ISSUE #3

NOVEMBER 1975

BYTE

$1.50
(12 bits)

the small systems journal

A $20 Microprocessor ?

Burn Your Own ROMs

Computer Hams ?

Ins and Outs of Volatile Memories

Computers Are Ridiculously Simple !

Is This Next ?



Computers -

The World's Greatest Toy !

The MODULAR MICROS from MARTIN RESEARCH

Here's why the new *MIKE 2* and *MIKE 3* are the best values in microcomputers today!

8008 OR 8080

Martin Research has solved the problem bothering many potential micro users .... whether to go with the economical 8008 microprocessor, or step up to the powerful 8080. Our carefully designed bus structure allows either processor to be used in the same system!

The *MIKE 3* comes with an 8080 CPU board, complete with crystal-controlled system timing. The *MIKE 2* is based on the 8008. To upgrade from an 8008 to an 8080, the user unplugs the 8008 CPU board and plugs in the 8080 CPU. Then he unplugs the 8008 MONITOR PROM, and plugs in the 8080 MONITOR PROM, so that the system recognizes the 8080 instruction set. That's about it!

If the user has invested in slow memory chips, compatible with the 8008 but too slow for the 8080 running at full speed, he will have to make the 8080 wait for memory access—an optional feature on our boards. Better still, a 4K RAM board can be purchased from Martin Research with fast RAM chips, capable of 8080 speeds, at a cost no more that you might expect to pay for much slower devices.

In short, the *MIKE 2* user can feel confident in developing his 8008 system with expanded memory and other features, knowing that his *MIKE 2* can be upgraded to a *MIKE 3*—an 8080 system—in the future.

EASE OF PROGRAMMING

Instructions and data are entered simply by punching the 20-pad keyboard. Information, in convenient octal format, appears automatically on the seven-segment display. This is a pleasant contrast to the cumbersome microcomputers which require the user to handle all information bit-by-bit, with a confusing array of twenty-odd toggle switches and over thirty red lights!

A powerful MONITOR program is included with each microcomputer, stored permanently in PROM memory. The MONITOR continuously scans the keyboard, programming the computer as keys are depressed.

Say the user wishes to enter the number *135* (octal for an 8008 OUTPUT 16 instruction). He types *1,* and the right-hand three digits read *001.* Then he presses *3,* and the digits say *013.* Finally he punches the *5,* and the display reads *135.* Notice how the MONITOR program

(Continued in column 3.)



shifts each digit left automatically as a new digit is entered! The value on the display is also entered into an internal CPU register, ready for the next operation. Simply by pressing the *write* key, for example, the user loads *135* into memory.

The MONITOR program also allows the user to step through memory, one location at a time (starting anywhere), to check his programming. Plus, the Swap Register Option allows use of the interrupt capabilities of the microprocessor: the MONITOR saves internal register status upon receipt of an interrupt request; when the interrupt routine ends, the main program continues right where it left off.

We invite the reader to compare the programmability of the *MIKE* family of microcomputers to others on the market. Notice that some are sold, as basic units, *without any memory capacity at all.* This means they simply cannot be programmed, until you purchase a memory board as an "accessory." Even then, adding RAM falls far short of a convenient, permanent MONITOR program stored in PROM. Instead, you have to enter your frequently-used subroutines by hand, each and every time you turn the power on.

EASY I/O INTERFACE

The *MIKE* family bus structure has been designed to permit easy addition of input and output ports. A hardware interface to the system generally needs only two chips—one strobe decoder, and one latching device (for output ports) or three-state driving device (for inputs). A new I/O board can be plugged in anywhere on the bus; in fact, all the boards in the micro could be swapped around in any position, without affecting operation. I/O addresses are easy to modify by reconnecting the leads to the strobe decoder (full instructions are provided); this is in marked contrast to the clumsy input multiplexer approach sometimes used.

POWER & HOUSING

The micros described to the left are complete except for a cabinet of your own design, and a power supply. The basic micros require +5 V, 1.4 A, and −9 V, 100 MA. The 4K RAM board requires 5 V, 1 A. A supply providing these voltages, and ±12 V also, will be ready soon.

OPTIONS

A number of useful micro accessories are scheduled for announcement. In addition, the *MIKE 3* and *MIKE 2* may be purchased in configurations ranging from unpopulated cards to complete systems. For details, phone, write, or check the reader service card.

RGS ELECTRONICS

DISCOUNTS: 10% OFF ORDERS OVER $25.00; 20% OFF ORDERS OVER $250.00.

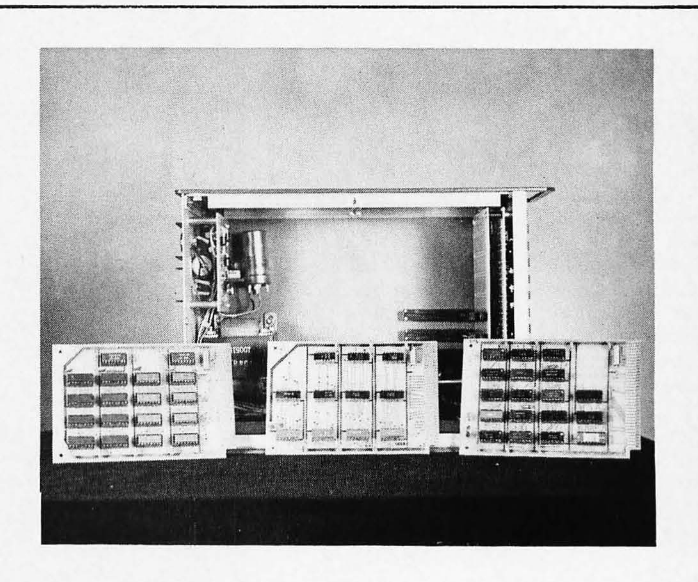
SPECIAL
1-8008
8-2102
$50

ANOTHER POWER SUPPLY . . .

PS 25—1 0 to 25v 1a lab type power supply with adjustable current limiting; remote sensing & remote programming for voltage & current. Insturctions included. All parts except chassis, meter(s), p.c. board.
Kit of parts with schematics. $14.95
P.C. boards available, No. 007 $3.00 ea.

2K RAM BOARD KIT. ALL PARTS INCL. SOCKETS

$84.50



ICs

8008	MICROCOMP. CHIP	$30.95
2102	1K STATIC RAM	3.00
5203	256x8 PROM	15.00
5204	512x8 PROM	25.00

INFO ON ABOVE CHIPS IF ASKED FOR.

008A MICROCOMPUTER KIT

8008 CPU, 1024 x 8 memory; memory is expandable. Kit includes manual with schematic, programming instructions and suggestions; all ICs and parts supplied except cabinet, fuses & hardware. Includes p. c. boards. $375.00

MANUAL ONLY, $25.00
(no discount on manual)

ORDERS OF $50 OR MORE GET FREE BYTE SUBSCRIPTION IF ASKED FOR (CONTINENTAL U.S. ONLY).

008A-K ASCII keyboard input kit. $135.00

008A-C Audio cassette adapter kit. $100.00

Details on computer, peripheral kits in our flyer.

RGS ELECTRONICS

3650 Charles St., Suite K ■ Santa Clara, CA 95050 ■ (408) 247-0158

We sell many ICs and components not listed in this ad. Send a stamp for our free flyer. TERMS OF SALE: All orders prepaid; we pay postage. $1.00 handling charge on orders under $10.00. California residents please include sales tax. Please include name, address and zip code on all orders and flyer requests. Prices subject to change without notice.

In the Queue

Foreground

Background

Nucleus

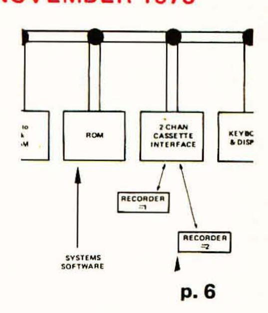
BYTE magazine is published monthly by Green Publishing, Inc., Peterborough, New Hampshire 03458. Subscription rates are $12 for one year worldwide. Two years, $22. Three years, $30. Second class postage application pending at Peterborough, New Hampshire 03458 and at additional mailing offices. Phone: 603-924-3873.

In case you were wondering, last month's cover had pictures of several MITS and SWTCP computer kit boards.

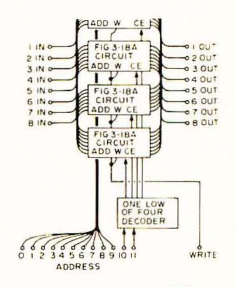
Contract:
The subscriber or purchaser of this magazine agrees to the following software conditions . . . not to resell this magazine for less than 50% of the cover price . . . not to give the magazine away at any time in the future . . . or to lend it . . . or rent it or in any other way permit anyone to become privy to the material published within these pages. Purchaser agrees to display this copy of BYTE to as many computer addictees as possible, but to limit their perusal to the cover and table of contents pages. This agreement holds not only for casual acquaintances, but also for personal friends, blood relatives, and even wives.

BYTE #3

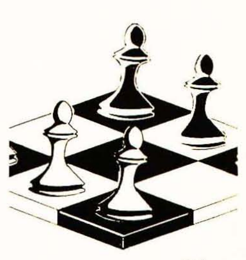
NOVEMBER 1975



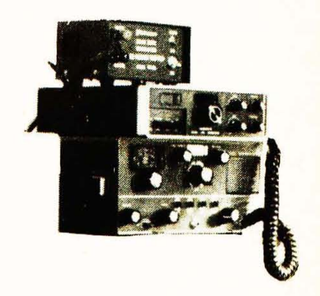
p. 6



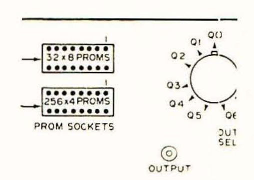
p. 12

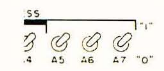


p. 36



p. 42





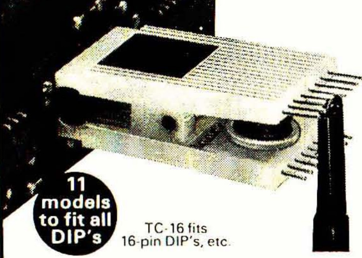


p. 66

COVER: See page 5.

IC troubles? try these high-performance Super-Grip™ IC Test Clips

for fast, non-shorting access to all leads on dual-in-line IC packages



11 models to fit all DIP's

TC-16 fits 16-pin DIP's, etc.

Model	Row-To-Row Dim.	Part Number	Price
TC-8	.3 IN.	923695	$7.35
TC-14	.3 IN.	923698	4.50
TC-16	.3 IN.	923700	4.75
TC-16 LSI	.5/.6 IN.	923702	8.95
TC-18	.3 IN.	923703	10.00
TC-20	.3 IN.	923704	11.55
TC-22	.4 IN.	923705	11.55
TC-24	.5/.6 IN.	923714	13.85
TC-28	.5/.6 IN.	923718	15.25
TC-36	.5/.6 IN.	923720	19.95
TC-40	.5/.6 IN.	923722	21.00

No more shorting across DIP leads... just quickly clip on an IC TEST CLIP to bring DIP leads out for safe attachment of scope probes and other leads. Ideal for signal input, tracing, troubleshooting, etc. Patented precision, "contact comb" design guarantees no shorting between DIP leads. Probes can hang "no-hands" free on Test Clip terminals in card racks (unique — see photo). Engineered Mechanical clamping plus gold-plated phosphor bronze terminals provide superior electrical contact. *Also unequaled as a DIP removal tool.*

We honor M.C. and B.A.C. charges. Add sales tax on OH and CA orders. (F.O.B. Painesville on company P.O.'s.) *Dealer inquiries invited*

Add fees from this chart.

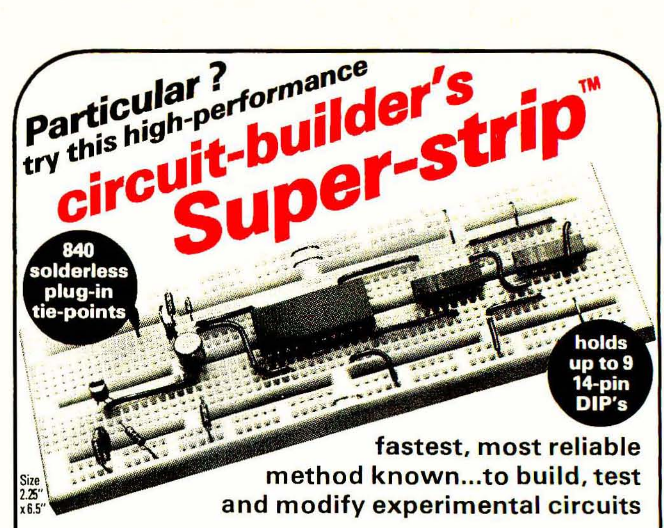
SHIPPING/HANDLING		C.O.D.
Up to $10.00	$1.00	$ .70
$10.01 to $25.00	1.50	.80
25.01 to 50.00	2.00	.90

All products guaranteed to meet or exceed published specifications



A P PRODUCTS INCORPORATED
Box 110-G • Painesville, OH 44077 • 216/354-2101

Particular? try this high-performance circuit-builder's Super-strip™



fastest, most reliable method known...to build, test and modify experimental circuits

8-bus distribution system plus universal .1" by .1" matrix of solderless, plug-in tie points all in one integral device.

Accepts all DIP's and discretes with leads up to .032" dia.

Interconnect with any solid wire up to No. 20 A.W.G.

ORDER BY PART NUMBER
923252 (nickel-silver terminals)....... **$17.00**
923748 (gold-plated terminals)........ **$18.90**

High-performance A P Super-Strip component matrices consist of 128 terminals of 5 tie points each. The 8 buses of 5 connected 5-tie-point terminals are for voltage, ground, reset and clock lines, shift command, etc. New, non-shorting, instant-mount backing and quick-removal screws are supplied... use several Super-Strips to create large-scale breadboards on panels up to 1/8" thick.

We honor M.C. and B.A.C. charges. Add sales tax on OH and CA orders. (F.O.B. Painesville on company P.O.'s.) *Dealer inquiries invited*

Add fees from this chart.

SHIPPING/HANDLING		C.O.D.
Up to $10.00	$1.00	$ .70
$10.01 to $25.00	1.50	.80
25.01 to 50.00	2.00	.90

All products guaranteed to meet or exceed published specifications



A P PRODUCTS INCORPORATED
Box 110-G • Painesville, OH 44077 • 216/354-2101

Creative? try these high-performance circuit-builder's Terminal and Distribution Strips

All solderless plug-in tie-points

Super-Versatile™ building blocks for experimental circuits

Universal .10" matrices of solderless, plug-in tie points

For all DIP's and discretes with leads to .032" dia.

Interconnect with any solid wire up to No. 20 A.W.G.

Nickel-silver terminals

ORDER BY PART NUMBER
923277 distribution strip . . . **$2.50**
923261 terminal strip **$12.50**

Create custom breadboards in minutes with these new instant-mount strips. DISTRIBUTION STRIP (top) contains 2 continuous buses of 12 connected 4-tie-point terminals. Size: 6.5" by .35". TERMINAL STRIP contains 128 5-tie-point terminals, holds up to *nine* 14-pin DIP's. Size: 6.5" by 1.36". Integral, non-shorting, instant-mounting backing permits quick build-up of special breadboards using any mix of strips.

Other models available.

We honor M.C. and B.A.C. charges. Add sales tax on OH and CA orders. (F.O.B. Painesville on company P.O.'s.) *Dealer inquiries invited*

Add fees from this chart.

SHIPPING/HANDLING		C.O.D.
Up to $10.00	$1.00	$ .70
$10.01 to $25.00	1.50	.80
25.01 to 50.00	2.00	.90

All products guaranteed to meet or exceed published specifications



A P PRODUCTS INCORPORATED
Box 110-G • Painesville, OH 44077 • 216/354-2101

In a hurry? try these high-performance A·C·E All-Circuit Evaluators

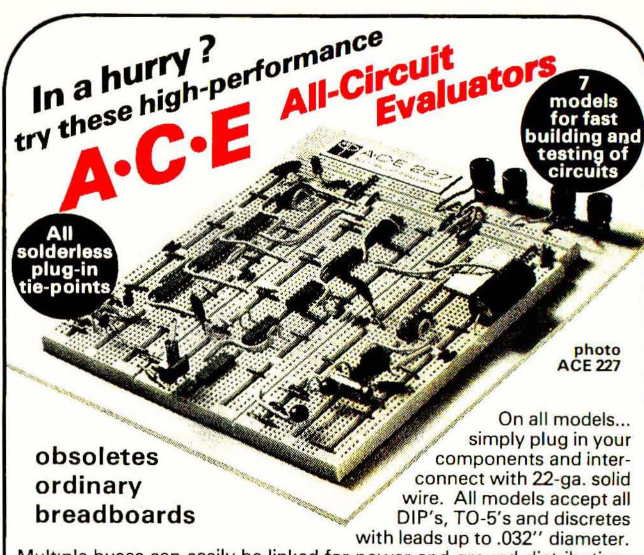


photo ACE 227

obsoletes ordinary breadboards

On all models... simply plug in your components and interconnect with 22-ga. solid wire. All models accept all DIP's, TO-5's and discretes with leads up to .032" diameter. Multiple buses can easily be linked for power and ground distribution, reset and clock lines, shift command, etc. Bases: gold-anodized aluminum. Terminals: non-corrosive nickel-silver. Four rubber feet included.

Order No.	ACE Model No.	Tie Points	DIP Capacity	No. Buses	No. Posts	Board Size (inches)	Price Each
923333	200-K (kit)	728	8 (16's)	2	2	4-9/16 x 5-9/16	$18.95
923332	208 (assem.)	872	8 (16's)	8	2	4-9/16 x 5-9/16	28.95
923334	201-K (kit)	1032	12 (14's)	2	2	4-9/16 x 7	24.95
923331	212 (assem.)	1224	12 (14's)	8	2	4-9/16 x 7	34.95
923326	218 (assem.)	1760	18 (14's)	10	2	6-1/2 x 7-1/8	46.95
923325	227 (assem.)	2712	27 (14's)	28	4	8 x 9-1/4	59.95
923324	236 (assem.)	3648	36 (14's)	36	4	10-1/4 x 9-1/4	79.95

We honor M.C. and B.A.C. charges. Add sales tax on OH and CA orders. (F.O.B. Painesville on company P.O.'s.) *Dealer inquiries invited*

Add fees from this chart.

SHIPPING/HANDLING		C.O.D.
Up to $10.00	$1.00	$ .70
$10.01 to $25.00	1.50	.80
25.01 to 50.00	2.00	.90

All products guaranteed to meet or exceed published specifications





A P PRODUCTS INCORPORATED
Box 110-G • Painesville, OH 44077 • 216/354-2101

from the Publisher . . .

Aggravated shock. How else can I describe my state of mind after innocently turning my newly acquired Altair 8800 around, looking for places to plug in Teletypes and television typewriters.

After forty years of turning around electronic equipment to plug things in, this time there was nothing there except the line cord. Nothing! Not even a lousy phone jack.

High fidelity amplifiers have the controls and switches on the front panel and the terminals and jacks on the back like any decent piece of equipment ought. As a matter of fact, some of the amplifiers these days have just about gone berserk with jacks and terminals. I had no problem plugging in my stereo phono and the tape recorder, but I haven't located any source of auxes as yet for that line. It's in stereo too. Know anyone with a good stereo aux for sale – cheap?

Ham transmitters and transceivers have lots of places to plug things in all over the back panel. I notice, upon checking, that some of them have a place for my aux, if I manage to get one. It'll get a lot of use.

Maybe the jacks are inside, I thought. So I unscrewed the lid and slid it back – still no terminals, no jacks, no plugs, no connectors, nothing! In desperation I grabbed the instruction book. It was a thick one so I moved to a comfortable chair, adjusted the light, and started reading about gates, truth tables, and I fell asleep. You wouldn't believe the crook in my neck that chair gave me when I tried to get out of it the next morning.

Now, after having read as far as I could in the instruction book and after talking with a number of experts and even visiting the MITS factory, I am beginning to suspect that I am going to have to do a good deal more than just plug in a Teletype machine to hook into my Altair. Interface is the buzz word. It means not only plugging it in, but doing so in a way that will enable it to work.

My Teletype machines have distributors built-in, which means that the Beau's Dots come out one after the other rather than all at once

Continued on page 96



The wave of the future? This is a shop in a mall in Dallas.

BYTE staff

EDITOR
Carl T. Helmers Jr.
PUBLISHER
Wayne Green
MANAGING EDITOR
Judith Havey
ASSOCIATE EDITORS
Dan Fylstra
Chris Ryland
CONTRIBUTING EDITORS
Hal Chamberlin
Don Lancaster
ASSISTANT EDITORS
John C. Burnett
Susan G. Philbrick
PRODUCTION MANAGER
Lynn Panciera-Fraser
ART DEPARTMENT
Nancy Estle
Neal Kandel
Peri Mahoney
Bob Sawyer
PRINTING
Biff Mahoney
PHOTOGRAPHY
Bill Heydolph
Ed Crabtree
TYPESETTING
Barbara J. Latti
Marge McCarthy
ADVERTISING
Bill Edwards
Nancy Cluff
CIRCULATION
Susan Chandler
Dorothy Gibson
Pearl Lahey
Lorraine Morton
Judy Waterman
INVENTORY CONTROL
Marshall Raymond
DRAFTING
Bill Morello
COMPTROLLER
Knud E. M. Keller

Carl Helmers

The State of The Art

If there is one facet of the small computer field which is its most exciting, that is probably its rapid change and evolution unfolding before all us users of the technology. The fact that a magazine such as BYTE can even exist (let alone get its enthusiastic reception) is evidence of the considerable changes which have occurred in the home computer field over the past year or two. Any attempt such as this to characterize the current "state of the art" is doomed to rapid obsolescence. Be that as it may, I won't let that deter me from characterizing the field as I see it now.

Just what is this "art" that I'm talking about? When I talk about art in this sense, I mean the body of technological know-how available for personal computing plus the attitudes and abilities of the people who use this know-how. An analogy or two: The state of the art in a form such as painting reflects both the latest developments in the pigmentation materials field and the creative talents and attitudes of the people who use this technology for expressive purposes. The state of the art in music is a combination of the technology of music production – traditional to electronic/digital – plus the aesthetic and creative tastes of the musicians and composers who use the technology. So it is as well with computing. There is the technological state of the art as it exists – a transient thing at present – together with the creative uses to which people such as you or I put these wonderful technological devices.

Any attempt to specify the state of the art in this field is doomed to practically instant obsolescence . . .

A Recent State of the Art . . .

A few years ago, the state of the art in hardware was pretty primitive – in other words, one had to be a really persevering person to get something in computing which worked and cost less than $1000. To give you an example, I got a call from Dick Snyder of Chelmsford, Mass., shortly after BYTE #1 came out. (See Dick's letter in the letters column of this issue.) As a result of our conversation, I stopped at Dick's house on the way back from Peterborough one weekend in August and took a look at his pre-microcomputer home brew computer, a really beautiful piece of work. He had completely designed and built – in 1972 and 1973 – a miniature 4-bit computer with 256 nybbles of memory using the Data General NOVA minicomputer as his inspiration. He built the machine using painstakingly accurate soldering with a miniature iron, sockets for over 170 integrated circuits, and a very compact housing. The most unusual feature of all was the use of water cooling to keep his 16 7489 memory chips cool (said water cooling consisting of plastic bag baby bottles filled with water and sealed with rubber bands). Yet it works! And – he has built up quite an impressive array of software for his one-of-a-kind machine, including a very appealing simulation of a priority-driven real time operating system with three tasks in the queue. The entire program for this simulation is done in 256 nybbles (half-bytes) of memory with the 16 instructions of his design. The result is an impressive changing display of marker patterns in his front panel lights as the various tasks swap in and out of execution. Dick Snyder's machine is the state of the art, circa 1972-1973, to a large extent – microcomputers were not yet widely available to the general populace of personal computing hackers. Dick tells me that he spent about $600 on the parts of his computer at 1972 prices for SSI and MSI TTL integrated circuits.

But now, in 1975 after the first wave of 8008 computer kit products and the rising tide of the "first generation" personal computer systems, that same $600 can buy a lot more function. In 1975 we saw the introduction of the MITS ALTAIR – which turns out to be a very good computer after initial slow deliveries due to unanticipated demand – and a host of new machines such as Bill Godbout's PACE, the SWTPC 6800 kit, the MITS 6800 kit and several other systems.

The Benchmark of a Small Computer System

In the engineering and software professions, it is often common to dream up "benchmarks" to help in the evaluation of systems. This term, benchmark, was adopted by systems engineers from its original use in the field of geodetic surveying. A geodetic survey benchmark is a permanent marker set "out in the field" (literally) at known locations during the course of the survey. If you clamber to the top of Mt. Chocorua in New Hampshire, as I sometimes do, when you get to the top you will find a little metal plate giving elevation, longitude and latitude information. This is the benchmark for the mountain's peak. Well, the benchmarks used for computer systems are a little bit less concrete than a metal plate on a mountaintop, but serve the same purpose: They provide a reference point for comparison.

A common benchmark which has been used in the past to evaluate computer systems (and compilers) is the "standard set of programs". In this method of benchmarking a system, the potential user of the system picks a set of "typical" applications programs and has them implemented and measured in operation on several different systems. This is a fairly quantitative and seemingly accurate method which is widely practiced in the information systems

industries. The measurements made for comparison include "through-put" (processing per unit time), high level language efficiency, memory requirements, etc. But this sort of a measure is perhaps a bit too complicated for the home computer context. For one thing, the applications are known only generally. Second, this is the type of study which takes a large amount of time and access to various competitive systems. And, if you read the trade journals, the results are often controversial anyway, since each manufacturer will claim that the benchmarks he provides will prove his machine better than all the rest. Picking the "ideal" small computer system still requires a benchmark, but I suggest it is not a particular program, but a *capability*.

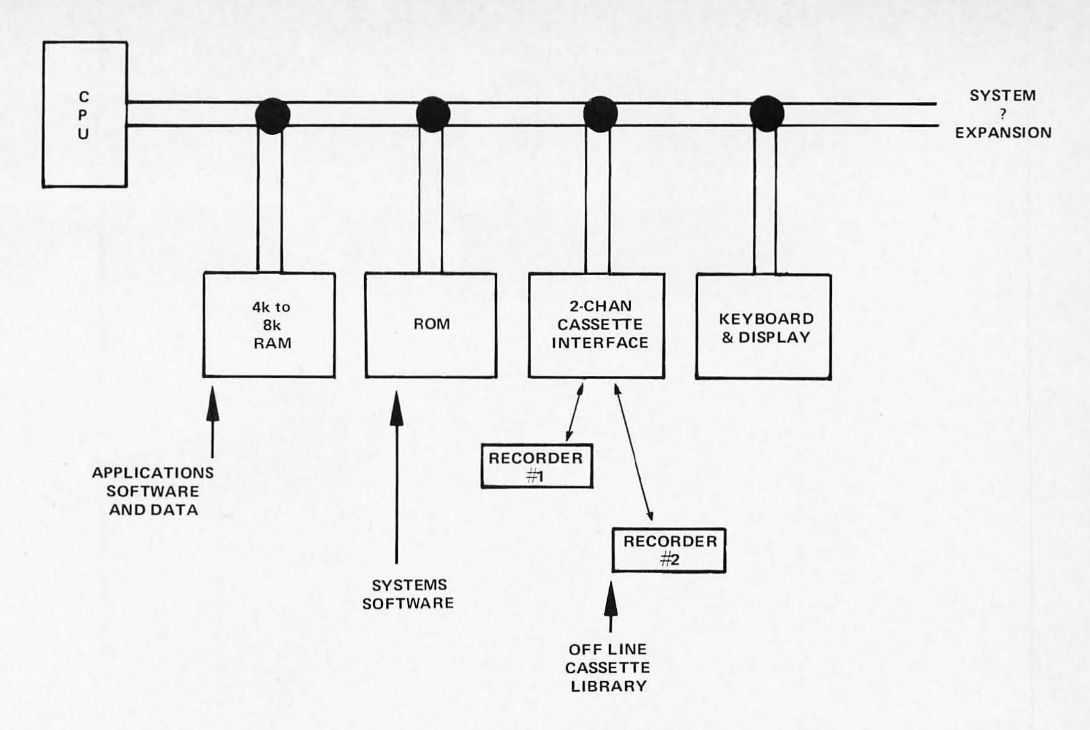


Fig. 1. The Complete Low Cost Computer System (circa September 1975). This diagram shows the major components of a typical low cost computer system – which should total up under $1000 depending upon manufacturer and details of design. At the time this editorial is written, several kit manufacturers meet this functional benchmark at prices well under $1000. As time goes on the improvements of mass production should drop the average price of such systems.

Capability – the Benchmark of a Small System

We all know that in broad terms, the benchmark computer system, as any computer system, must include several major components: a processor, memory, a mass storage medium, an interactive operator's terminal and systems software. I pick this list in part to illustrate a typical computer configuration and in part to allow programming of a benchmark capability:

> A small computer system which meets the benchmark standard will be able to *interactively edit* a mass storage file of input data with operator commands, producing a second mass storage file as output. This will be achieved in a system costing at most $1000 initially.

The system diagram of the benchmark computer is shown in Fig. 1, as it is implemented in the current state of the art. The components of the system are chosen with the editing function in mind, since accomplishing such an edit capability means the machine can be programmed for almost any other personal computing use. Peripherals that enhance the function are of course desirable and will help to personalize your system, but these functions represent the bare minimum without added cost of special purpose peripherals.

The CPU: Which One?

In Hal Chamberlin's article in BYTE #1, the relative merits of three computer designs were covered. In BYTE #3, Dan Fylstra covers a comparison of two additional designs. There is a large variety in the types of CPUs available to home brewers and kit builders – ranging from the 8008, 8080, 6800 and 6501 8-bit micros, to the 16-bit IMP and PACE micros, to commercial 16-bitters such as the LSI-11 and NAKED Milli products – and on into the never-never-land of custom designed microcoded MSI computers implemented by individuals (and also soon to be announced in product form by one manufacturer of kits). There is a large element of personal taste involved in the preference of particular instruction sets, and there is also the matter of efficiency for particular classes of programs. Whatever the CPU you use, it is a definite requirement of the system. I guarantee you that any one of the 8-bit or 16-bit microprocessors currently being packaged and sold as kits will be adequate to pass this benchmark test, although you may have to write the Editor program yourself.

RAM Memory – How Much?

The CPUs of the conventional microprocessors – kit or home brew implementations – create an output called a "data bus"

Picking your ideal computer system requires a benchmark – which I suggest is not a particular program but a *capability*.

which is used for exchanging information with everything else in the system. The data bus is the "spinal cord" of the computer's nervous system. This bus concept typically includes 16 bits of buffered address lines and several bus control information lines *as well as* the 8 or 16 bi-directional buffered data lines. The address space of the typical contemporary micro-computer's architecture is usually 16 bits worth or 65,536 possible memory locations. In the usual system most of these locations will

Continued on page 88

You'll get more than a core.



We make the difference count.

We'd like to be modest, but we've worked hard to bring about the lowest-cost computer systems on the market today; and it was no accident either. The Sphere team introduced really innovative micro-circuit design which brought big computer capabilities down to micro-processer size. You can sit down to a Sphere System and it will do the job you want to get done for personal or business use — from games to building security system or even to maintaining accounts receivable, accounts payable, and printing out checks, mailing lists and invoices. This is possible because your SPHERE system is totally expandable to use an extended BASIC compiler and/or the many high-quality, low-cost peripherals on the market today (also available from SPHERE). We provide manuals for maintenance and kit assembly, and even programming concepts for the first-time computer user.

Note our special offers on the following pages and send the coupon with your order or request for more information on our parts and kits, along with a complete line of cost-efficient peripherals. We will deliver in 60 days. At SPHERE CORPORATION you'll get a lot more than just a core . . . intelligent systems.

We make the difference count.

SPHERE CORPORATION
791 South 500 West # 1
Bountiful, Utah 84010
(801) 295-1368

Dear SPHERE:

Yes, I want more than a core. Tell me everything; send your detailed information.

Print Name: __________

Address: __________

City and State: __________

Zip Code: _____ Phone: _____



SPHERE CORPORATION 791 South 500 West # 1
Bountiful, Utah 84010 (801) 295-1368

4K Computer!



Still, our price goes down a lot easier.

Now's your chance to bite into a complete computer system. We at SPHERE CORPORATION have used the latest micro-processing technology along with some real innovations in mini-circuit design to develop the lowest-cost complete computer systems available. Every system comes complete with TV CRT display, ROM monitor, a real time clock, and typewriter, cursor editing, and numeric keyboards. Every SPHERE SYSTEM has ample memory to run YOUR particular program, whether it's a complicated inventory control system, or keeping detailed track of finances, or monitoring and protecting your home or office . . . use your imagination! And SPHERE offers manuals with plenty of information for the first-time computer users. In addition, every SPHERE SYSTEM is expandable as desired. It will run an extended BASIC compiler and/or a most complete line of peripherals. All systems come with complete usage, programming, and application manuals, as well as software supports to make its functions almost limitless. We wanted to make something a lot better; and we did!

The mail order offer you see here is limited. Take advantage of it! If you like, also inquire about our full color graphics CRT display, and SPHERE's most complete line of low-cost peripherals on the market today.
Use the coupon below right away!

We make the difference count.

SPHERE CORPORATION
791 South 500 West # 2
Bountiful, Utah 84010
(801) 295-1368

Dear SPHERE:

The facts I read here convinced me!

☐ Enclosed is my check or money order for $860.00. Please rush me a SPHERE I System.

☐ Please rush me more details on your low-cost computer system and peripherals. (Specific questions, if any, are attached.)

Print Name: ____________

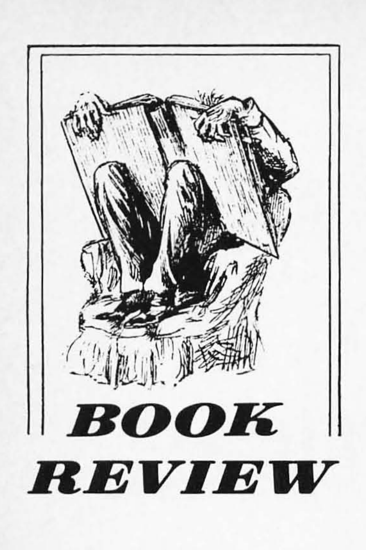
Address: ____________

City: ____________

State, Zip: ____________



SPHERE CORPORATION 791 South 500 West # 2
Bountiful, Utah 84010 (801) 295-1368



What To Do After You Hit Return, or PCC's First Book of Computer Games. *Available from People's Computer Company, PO Box 310, Menlo Park CA 94025 ($6.95 plus 50¢ postage and handling), or from the Hewlett-Packard Corporation, Mail Order Dept., PO Drawer #20, Mountain View CA 94043 ($6.95 plus $1.50 postage and handling).*

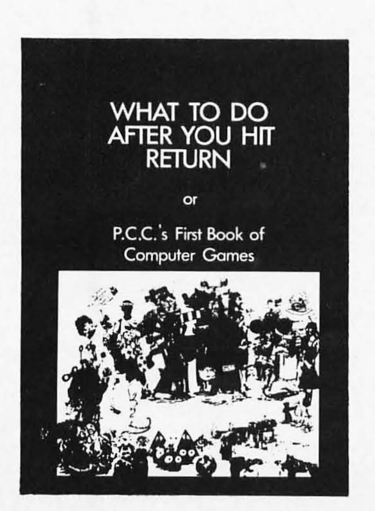
If you've managed to get BASIC up and running on your Altair or other home brew computer system, why not try it out with a few computer games implemented as BASIC programs? Or if you and your minicomputer or time-sharing service have some time to spare . . .

What To Do After You Hit Return is a book of computer games published by the People's Computer Company with the collaboration of Hewlett-Packard Corporation. All of the game programs are written in HP 2000F BASIC, and should be fairly easy to adapt to other implementations of the BASIC language. The book includes listings of most of the programs, as well as information on how to order machine-readable copies of the programs (on paper tape or magnetic tape) at reasonable prices.

The book is nicely organized into 10 chapters, each dealing with a certain "kind" of game, such as a word game or a hide-and-seek game. The programs range from very elementary number-guessing games to sophisticated business and social science simulations (listings of the larger programs are not included in the book). I was somewhat disappointed to find that a listing of the very popular "Star Trek" game was omitted, although an elaborate "Star Trader" game is included.

The first few pages of the book present, among other things, a rationale for the existence of computer games. While this is probably an effective way to counteract the rigid, efficiency-minded attitudes of some computer professionals, it's really sufficient to say the obvious, "Games are fun". My only reason for hesitation in recommending this book is that I have known some people whose work and play revolve entirely around computers. Therefore I would also like to recommend at the same time: sex, sunshine, salt water, snow, speed (cars, motorcycles, gliders, skydiving, whatever you like), yoga and music. The list could be extended indefinitely, of course, and the point is simply to use the whole of your mind and body. Those of us who work and play with computers know that, for exercising one's reason and imagination, we have the greatest tools and toys ever invented. So, having fulfilled my duty to remind you that one can have too much of a good thing, I am pleased to recommend this book, one of several heartening developments that promise to show people what computers are really all about.

—d.h.f.



Microcomputer Design, *by Donald P. Martin, Martin Research Ltd., 1825 S. Halsted St., Chicago IL 60608, 1-312-829-6932. $50.*

This book is terrific, especially for the following people:

a) Anyone doing original circuit design with microcomputers.

b) Anyone using the 8008 beyond simple applications. The book is virtually an encyclopedia on the 8008. It also mentions 8080 differences, where relevant. There are over 300 well-written pages filled with pertinent technical details.

This book is an excellent reference for the hardware microcomputer designer. Software is not the focal point, although numerous subroutines are fully listed. It is obvious the author has designed, debugged and used everything about which he writes. Whew! After reading this book he has my utmost professional respect. Considering the dollar per hour fees paid for consultants, this book is a bargain.

Format of the book is:

a) description of the basic requirements and minimal circuitry to support an 8008

b) detailed descriptions of special topics, including: I/O port expansions beyond manufacturer's specifications, adding instructions to the 8008, simultaneous input and output with a single instruction, bus structures, RAMs and PROMs, a detailed application of computer controlled programming of the 74S288 which appears adaptable to the popular and low cost 74188A/8223 PROM, a comprehensive solution for priority interrupt hardware, UART interface, keyboard interface, a complete A/D interface design with both single and multi-channel, interval timers, digital displays, etc. The last two chapters of the book describe four microcomputer circuits for the 8008: a nine-chip microcomputer, a $20 microcomputer, a seven-chip microcomputer, and a 19-chip microcomputer.

Author Martin assumes only that a reader has knowledge of TTL. The book does not repeat information found in Intel manuals. It is well organized and does not try to snow. This book is easily read and well digested, except for the sheer quantity of information.

Martin Research Ltd. sells the book directly at $50 and also is manufacturing a new series of microcomputer hobbyist kits. Judging by the quality of the book they should be excellent and without compromise. A comment in the recent *Micro-8 Computer User Group Newsletter* appears to bear this up: "It's a neat little machine . . ."

John Gilchrist
PO Box 1087
Glen Burne MD 21067

John Gilchrist is an independent consultant with Microprocessors Unlimited: μPoo.

The Ins And Outs Of Volatile Memories

Don Lancaster provides us with this discussion of some of the read/write memory techniques which are available to experimenters using readily available parts. In this background tutorial, Don discusses memory techniques from the simple gate flip flop to the bus-oriented RAM system using static memory circuits. For more detailed looks at the designs of circuits using some of the techniques in this article, readers should turn to Don's book, The TTL Cookbook, *available from Howard W. Sams, Indianapolis IN. The material in this article is abstracted from Chapter 3 of Don's forthcoming* TV Typewriter Cookbook, *also to be published by Sams.*

by
Don Lancaster
Synergetics

Unfortunately, there is no cheap and reasonable memory system available today that will both remember information forever *and* be able to read and write information rapidly, cheaply, and with reasonable timing signals. This is called the volatility problem.

A memory is non-volatile if it remembers forever. A volatile memory loses its contents if you remove supply power or fail to observe any timing restrictions it might have.

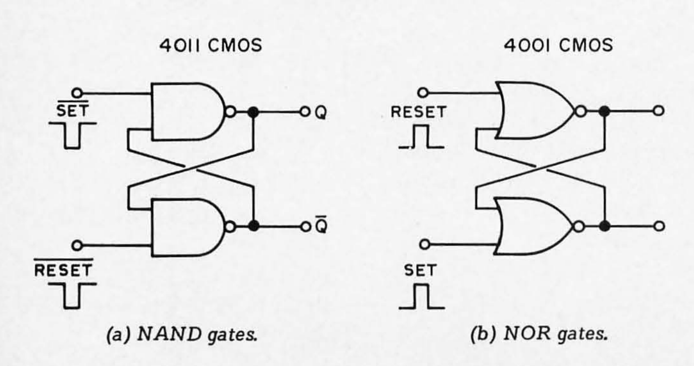
One older and obvious non-volatile memory system, of course, is magnetic core. The problem with core is that much in the way of support circuitry (including sense amplifiers, write after destructive read circuits, system timing generators, power down interrupts, etc.) makes core highly impractical for small scale TV typewriter and microcomputer systems.

Sometimes you can gain non-volatility with a volatile memory by some system level tricks, such as a power down technique that holds a low voltage from a battery applied to the memory when the main supply power goes away. Newer CMOS RAM memories consume almost negligible power in the standby mode and lend themselves well to this. You can also transfer data to some non-volatile outside storage such as a cassette recorder or a magnetic disc.

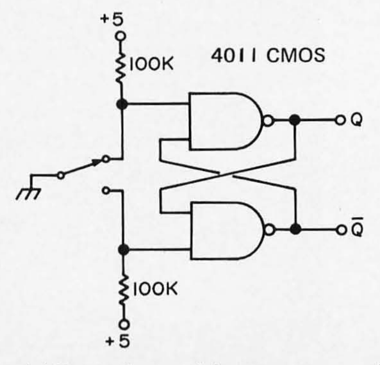
Read-write memory circuits can have their contents changed rapidly at system timing rates. This makes them useful for storing characters, computer programs, update commands, and anything else we want to temporarily store and recover. Most read-write memory circuits are volatile, holding their information only as long as supply power remains present and so long as any timing restrictions are not ignored.

Read-write memories for TVT and microcomputer use can range from single bit control and debouncing circuits up through thousand word character stores to 16k and 32k microcomputer memory stems. Some of the more important memory types include the set-reset flip flop, the word storage latch, the shift register, the Random Access Memory or static RAM, micropower RAMs, and dynamic RAMs. Let's look at these in turn.

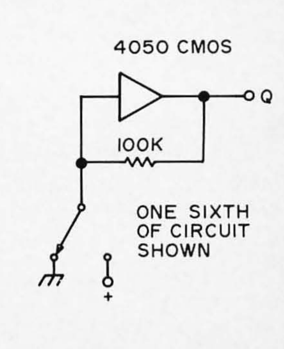
Fig. 1. Set-reset flip flops are the simplest type of read-write memory.



(a) NAND gates. *(b) NOR gates.*



(c) Bounceless push-button.



(d) Hex contact conditioning.

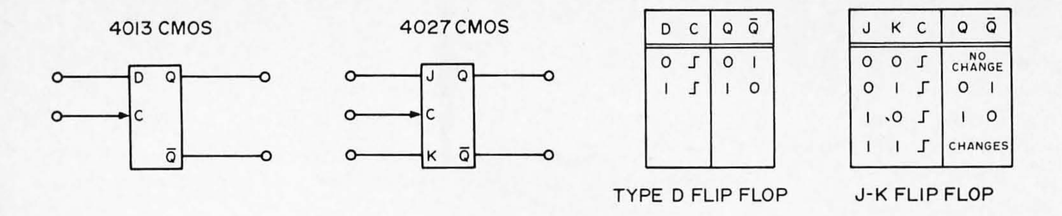
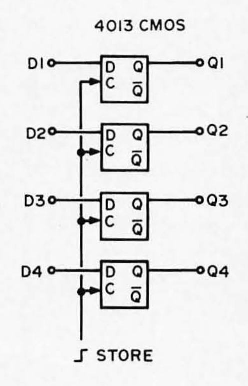
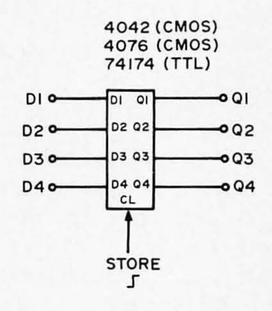


Fig. 2. Clocked flip flops only change in response to a control signal called the "clock".

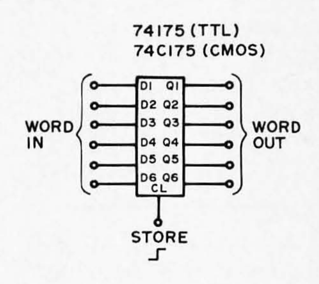
Fig. 3. Word storage latches provide a handy way to keep track of multiple bits of data.



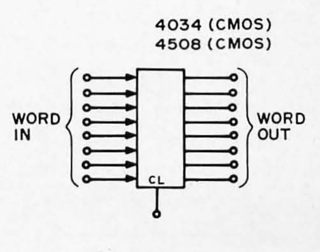
(a) 4-bit word using D-flops.



(b) 4-bit word using MSI.



(c) 6-bit word.



(d) 8-bit word.

The Set-Reset Flip Flop

One of the simplest read-write storage systems is the set-reset flip flop. It can store only one bit of information, and may be built using the NOR gates of Fig. 1(a) or the NAND gates of Fig. 1(b). When set, the Q output goes and stays in the "1" condition. When reset, the Q output goes and stays in the "0" state. A complementary or $\overline{Q}$ output is also supplied – it's a "1" when Q is a "0" and vice versa. Simple set-reset flip flops have a hangup in that if you simultaneously try to set and reset them, they go into a disallowed state, and the final way they end up depends on the last input to be released.

Fig. 1(c) shows us how to use a set-reset flip flop to eliminate the mechanical noise and bouncing of a SPDT push-button or mechanical contact. The first instant the contact makes, the flip flop jumps to the "1" state and stays there till the first instant after the switch is completely released, eliminating any bounce, noise or chatter. Circuits of this type are absolutely essential anytime you want to enter data from a switch or mechanical contact into any digital system. Fig. 1(d) shows us a simple hex contact conditioner using one CMOS non-inverting buffer.

A major use of set-reset flips flops is to debounce switches.

The Storage Latch

Operation of a set-reset flip flop is nearly instantaneous. If we tried to cascade a bunch of them, we'd get an unchecked wild race, for after changing the first one, the rest may follow domino style. It's much better to have digital circuits change only when you want them to and then do so on a one-stage-at-a-time orderly basis. To do this, we go to a clocked flip flop, such as the type D or JK flip flops of Fig. 2.

With these devices, the D input or the JK inputs set up what the flip flop is to do, but the actual change isn't carried out till a certain edge or level of the clock input happens. With a "D" flip flop we can clock in a "1" or clock in a "0", most often on the positive edge of the clock. We can also divide by two with a D flop by cross coupling the $\overline{Q}$ output to the D input, making the logic block change states every clocking. With the JK flip flop, we have the options of clocking in a "1", a "0", doing absolutely nothing, or changing each and every state as a binary frequency divider. For convenience in use, most JK and D flip flops also have extra direct set and reset inputs; these operate immediately and are useful for clearing or initializing memory states.

Word Storage

One flip flop can only store one bit at a time. If we use flip flops in groups, we can store a multiple bit word at once. In Fig. 3(a), we use four D flops to store a 4-bit word. Data is entered on the leading (positive) edge of the clock. Figs. 3(b), 3(c) and 3(d) show how we can use larger MSI logic blocks to store words of four, six and eight bits in length.

A word storage latch of this type is often handy to "catch" an input signal on the way by and hold it till you can use it. For instance, a microcomputer may output a word for only a microsecond or two, but your TVT circuit may not get around to using the word for milliseconds or even seconds later. In this case, you catch the microcomputer's output word with a storage latch and then keep it till you are certain you have used it. You then release the latch and ask the microcomputer for a new word through a handshaking signal.

Another important use for word storage is to resynchronize data and make sure it is valid. As an example, suppose your system changes words every microsecond, but that the

One bit of memory is one set-reset flip flop.

previous block that's giving you inputs takes 900 nanoseconds or so to get around to giving you a valid output after *its* inputs change. This only gives you 100 nanoseconds or so of valid data and may change with temperature or supply voltage. Add a storage latch and you can catch this output and hold it for the entire next microsecond. The output data will always be one microsecond late, but it will always be valid and always be locked to your system timing. A word storage latch of this type may be needed between the memory and the character generator of a TV typewriter if very long line lengths are in use.

Shift Registers

We can cascade a stack of type D flip flops so that they pass on their contents one stage to the right each clocking. This is called a shift register, and Fig. 4 shows several examples.

In Fig. 4(a), we've built a four stage register out of type D flip flops. Each clocking passes data one stage to the right. In Fig. 4(b), we've added some enter-recirculate logic to let us either send the data round and round or else change selected bits at once. We can make the register any length we like by adding extra internal storage flip flops.

A shift register is the digital equivalent of a "delay line"- - your bits go in now and come out later. An "electronic disk memory" is an extremely large memory made out of long shift registers - and is programmed by the computer almost exactly like an old-fashioned fixed head magnetic disk.

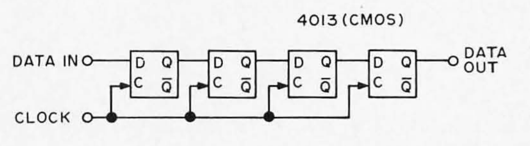
In Fig. 4(c), we use a MSI integrated circuit arranged as an 8-bit parallel-in-serial-out shift register. Shift registers are useful to convert dot matrix dots to serial video in a TVT; they are also handy for converting data from parallel to serial form. Fig. 4(d) is the opposite; this is a serial-in-parallel-out shift register useful to convert serial data into parallel form. Finally, Fig. 4(e) shows us a 1024-bit serial-in-serial-out register, useful for storing bulk data. Use several of these side by side if larger words are to be stored. For instance, to store 1024 ASCII characters, six of these could be used side by side. They can also be cascaded end to end for 1024, 2048, 4096 and more bits per word.

There are lots of long MOS shift registers available. Other popular bit arrangements include the hex 32-bit and hex 40-bit shift registers, the 2518 and 2519. On the surface, shift registers would appear to be ideal for character and program storage in TV typewriter circuits. One of the earliest TVTs (see September 73 *Radio Electronics*) made extensive use of shift register storage, and similar registers are still used in many premium computer terminals.

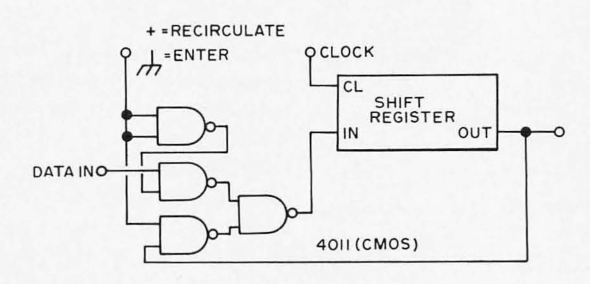
Today, we usually have a far better approach to data storage in the static random access memories of the next section. While these long shift registers were the first truly low cost semiconductor storage and are still useful for certain applications, they do have problems and the RAM techniques are often better.

Many of the early shift registers were dynamic devices in which you had to keep the data moving above some critical rate. Most early clocking circuits required a waveform that had to come from a fast, high current, noisy clock driver, swinging 17 volts or more with very strict pulsewidth and spacing

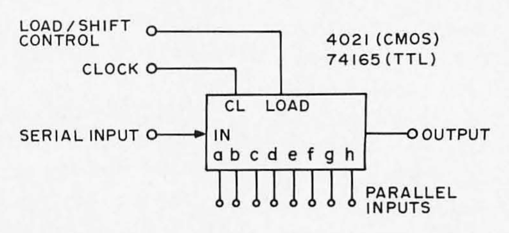
Fig. 4. Some shift register memories. Modern RAM memory devices tend to make shift registers obsolete except for special applications.



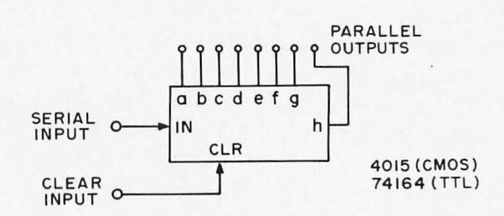
(a) Four stage using type D flip flops.



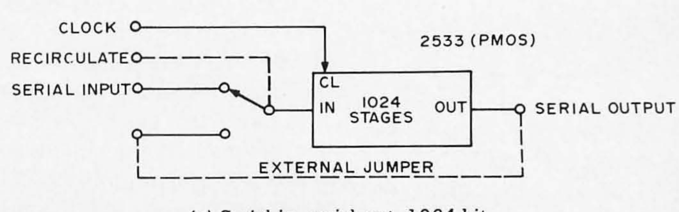
(b) Recirculate logic.



(c) Parallel in, serial out, 8-bit.



(d) Serial in, parallel out, 1024-bit.



(e) Serial in, serial out, 1024-bit.

Set-reset flip flops are asynchronous—in systems where outputs affect inputs, use of an SR latch can send you off to the races—an uncontrolled oscillation.

restrictions and any pulse overshoot strictly forbidden. Some earlier multiplexed shift registers, particularly the 1402, 1403 and 1404 exhibited selective dropouts called bit pattern sensitivity that would change data if the particular IC didn't happen to like the combination of clocking waveforms, supply voltage, temperature and internal data that it happened to have on hand.

Newer static N channel shift registers have eliminated most of these problems, but they still have one key drawback: This is simply that you can't immediately get at the data you want. The memory must be clocked around once and exactly once to get back any bit. All the other bits are usually between you and the bit you are after. With the RAMs of the next section you can selectively pick off any bit at any time, rather than waiting till it comes around. More importantly, you can be extremely sloppy about your timing and addressing between the times you are actually using the data, and the RAM doesn't care. With a shift register, one missed timing pulse is a disaster. Additional limitations of shift registers as bulk storage devices are that they often cost more at system levels than do RAMs and that they are not directly microprocessor compatible.

A bit is a bit. A word is "n" bits contemplated simultaneously.

Two areas where we can continue to see shift registers as important TVT and microprocessor parts are in First In First Out or FIFO buffer memories such as the Fairchild 3341 (64 x 4) and the 33512 (40 x 9) and similar devices by AMI and Western Digital, and in the newly emerging charge coupled device bulk storage systems such as the Fairchild CCD450 (1024 x 9) and CCD460 (128 x 32 x 4) devices. A FIFO gives us a way to interface two systems having different data rates, while the CCD devices promise to eventually provide very low cost and dense bulk storage.

Random Access Memory (RAM)

A random access memory or RAM differs from a shift register in that we can get to any memory location any time we want. If we like, we can address our RAM shift register style, working with the storage cells in sequential order. But we don't have to — we can get at any cell at any time in any order.

To do this, some external binary address lines are routed to internal decoder and selector circuits. When a memory cell is addressed, it is available either for reading as an output or for writing new information into it. Most RAMs tend to be only a single bit wide to save on package pins, but 4- and 8-bit words are sometimes offered.

There are lots of RAMs available. Bipolar types using TTL technology are usually fast and expensive. They generally provide less dense storage and fewer bits per package. Two examples are the 7489 arranged as 16 words of four bits each and the 74200 arranged as 256 x 1, or 256 words of one bit per word. Both cycle in under 50 nanoseconds.

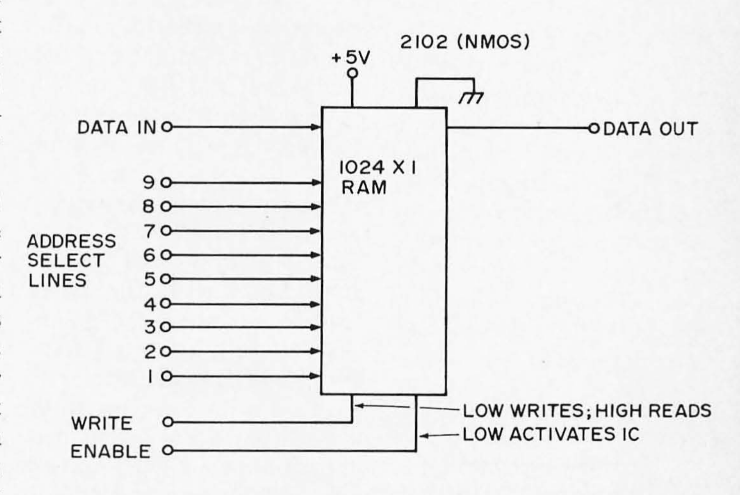
MOS memories using PMOS, NMOS and CMOS are also widely available. These are often slower and cheaper per bit. They usually offer more bits per package, with up to 4096 bits being common and 16384 just being talked about at this writing. Two early and essentially obsolete examples of MOS memory were the 1101 that was a fully static 256 x 1 device and the 1103 that was a dynamic RAM organized 1024 x 1 that singlehandedly toppled "king core" from the computer world. The early 1101s ran extremely hot with weird supply voltages, while the early 1103s had incredibly complex clocking, refresh, and timing restrictions, besides being bit pattern sensitive.

After several generations of much improved MOS memories, a device called the 2102 arrived and dropped enough in price to often be the best all around choice for TVT character storage and most smaller microcomputer memory tasks as well. The 2102 is arranged as 1024 x 1. It is N channel MOS and works on a single +5 volt supply and is fully compatible with TTL and CMOS logic on all pins. It is fully static, needing no clocks, refresh, charge pumps, memory busy interlocks, sense amplifiers, or similar garbage. Economy versions of the 2102 cycle in one microsecond, while premium jobs are available with 200 nanoseconds or less cycle time. Even the slowest 2102s are usually more than adequate for TVT use, although a bit slow for the newer microprocessors.

Best of all, at this writing, 2102s cost under $5 in singles and as low as $2 in large quantities. While designed by Intel, sources today include just about everybody — TI, Intersil, AMD, National, Signetics, Fairchild and Synertek.

Fig. 5 shows us the connections to a 2102 memory. We see a data input pin, a data output pin, 10 address lines, a write line, and an enable line. The data in

Fig. 5. The 2102 is the ideal RAM for many TVT and microcomputer uses.

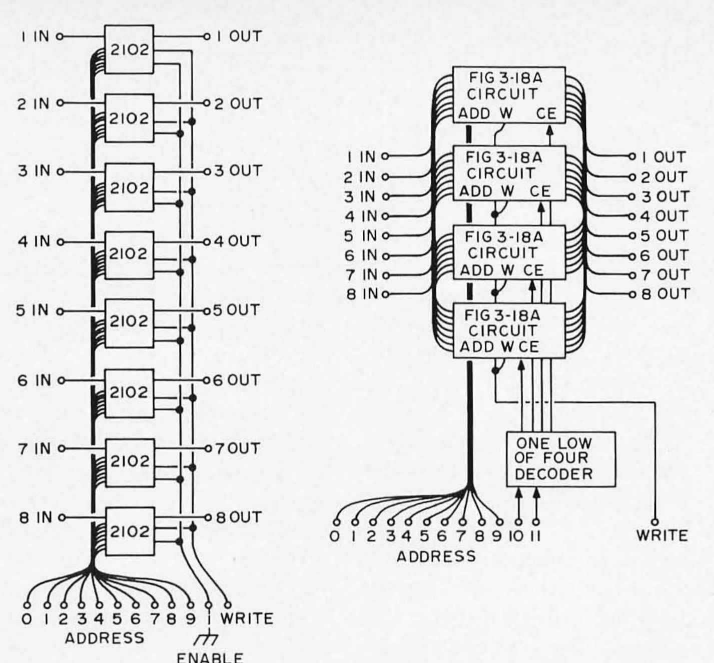


and data out lines are the same sense, meaning that a "1" input is stored as a "1" and appears as a "1" at the output. Our 10 address lines select one of the 1024 storage cells by providing binary addresses ranging from 00000-00000, 00000-00001 ... through 11111-11110 and 11111-11111. We can mix up input address lines any way we want so long as all packages and all input address timing circuits agree on what address combination goes with what storage cell. Jumbling the memory address inputs sometimes helps the circuit layout and makes such things as efficient BCD (Binary Coded Decimal) rather than binary addressing feasible.

Our chip enable controls whether the memory will do anything. If the chip enable is high, the data out line assumes a floating, high impedance, tri-state mode, and the write input logic is disabled. If the chip enable is low, the IC operates normally. *Chip enable should stay grounded except in very special uses.* The memory will read if the write input is high and will write if the write input is low.

There is one important timing restriction on the write input – *input addresses must be stable when the write input is low.* To write into memory, apply input data and an input address. After the inputs are stable, bring the write input low for at least the minimum write time. This time varies from 100 to 700 nanoseconds depending upon the device. Then release the write input, letting it go high before you change the address inputs or the data inputs. If you try to change addresses during the write process, certain memory locations may get "flashed" by in the address decoding and could lose or alter data. Incidentally this write-only-when-stable rule applies not only to 2102s – it should be observed with practically all RAM circuits. For normal 2102 read operation, make the write input *high* and the chip enable *low.*

Even the slowest 2101s are usually more than adequate for TVT use – although a bit slow for the newer microprocessors.



(a) 1024 x 8 character store memory.

(b) 4096 x 8 microcomputer memory.

Fig. 6. Larger 2102 memories are created by repeating the same circuit over and over.

Fig. 6 shows how we build a 1024 x 8 memory good for TVT character storage. Eight 2102s are simply put on the same PC card with their write, address and enable lines in parallel, the latter usually grounded. All inputs and outputs go to separate pins, and this way we can read or write 8-bit words. In Fig. 6(b), we've combined this circuit four times over to get a 4096 x 8 static memory suitable for a microprocessor main memory and big enough to hold a small compiler for an elementary higher level language. Two new address lines are one-of-four decoded and routed to the chip selects of each quarter of the memory. One chip select is made low and the other three remain high, and the tri-state outputs are shorted together as shown. A total of 32 ICs is needed.

When building either type of memory card, lots of 0.1 microfarad bypassing capacitors are recommended, along with wide supply and ground runs. PC layout is usually simplest with a double sided board and through-the-pins lead routing. Use plated through boards and keep the through-the-pins routing on the component side if possible.

Reorganized 2102s

Sometimes shorter words of more bits per word may be desirable. For these applications, some manufacturers have reworked the 2102 into different organizations including 128 8-bit words and 256 4-bit words. The Motorola and AMI 6810 are typical 128 x 8 units in a 24-pin package. The Signetics 2606 is a 256 x 4 version in a 16-pin package. Since there aren't enough pins to go around, the 2606 shares common input and output lines and must be used with a bidirectional data bus system. The Intel 2101 is a 22-pin version of the same thing with separate input and output pins, while the 2111 is an 18-pin memory that needs a bidirectional input/output system. At this writing, costs of these devices are considerably higher than conventional 2102s and are likely to stay that way since they have larger packages and less availability. Nevertheless, they often save you enough packages to be worthwhile in smaller systems. Having only 8 address bits makes the 256 x 4 memories directly compatible with 8-bit microprocessors as well.

Micropower Static RAMs

We can also build CMOS random access memories similar to the 2102. CMOS has one major advantage – its standby power needed when the memory is not cycling is almost zero. This makes CMOS memories ideal for "non-volatile" storage where a small battery can fill in for extremely long term data holding, as well as safely handling routine power outages. CMOS memory cells tend to be physically larger than NMOS ones and more process steps are often involved. So, CMOS memories will probably remain a more expensive route, but a very attractive one where micropower memory is essential. Obvious applications include electronic checkbooks and remote data acquisition systems.

Typical devices are the 64-bit Motorola 4505; 256-bit devices including the RCA 4061, Intersil 6523, and Motorola 4532 (the latter is arranged 64 x 4); 512-bit versions including the Nortec and AMI S2222, and the Intel 5105, arranged as 1024 x 1.

Traditionally, any particular size RAM starts out as an impossible to use dynamic device, upgrades itself into a difficult to use "quasi-static" device, and then gets replaced with a static no-hassle IC the third time around. Home brew computer people for the most part have to wait for the third stage or have a talent for wrestling with difficult hardware problems. At this writing, 1k memories are in stage 3, 4k memories are in stage 2, and 16k memories are at the beginning of stage 1.

Dynamic RAMs

A static RAM takes a full memory cell for data storage. We can get by with nothing but a capacitor as a storage device if we are willing to reshuffle, move around, or refresh the stored data more or less continuously. This is the principle behind the dynamic RAM. In exchange for cheap and dense storage, system level restrictions in the way of memory busy times, refresh cycles, clock lines, and clocking restrictions are needed, often combined with analog output sense amplifiers. Traditionally, any particular size RAM starts out as an impossible to use dynamic device, upgrades itself into a very difficult to use "quasi-static" device, and then gets replaced with a static no-hassle IC the third time around.

Because of this, dynamic RAMs should be avoided entirely for all TVT and microcomputer usage. While there are a wide variety of yet unstandardized 4096 x 1 dynamic RAMs on the market, including the Electronic Arrays 1504, Intel 2107, Standard Microsystems 4412, TI 4030, Mostek 4096, and the Motorola 6605, they presently cost much more than the equivalent storage using 2102s and are harder to get and harder to use. They do have the potential advantage of reducing package count 4:1 in very large memory systems where the 4096 x 1 format can be used, and are ideal for larger computer memories.

Bus Organization

Any memory system has input data lines, output data lines, and address data lines. It's usually simplest to keep these lines completely separate, for this way there are no timing commands needed to separate input, address and output signals, and no times can occur when one would interfere with the other. This is called an isolated bus or separate I/O system.

Many microcomputers instead use bidirectional data bus arrangements to save on pins and interconnections. The data bus just sits there. If something wants to transmit, its tri-state output is enabled. If something wants to receive, its input is enabled. The signals can go either way on the bus, but system timing has to make certain that only one source is transmitting and only the receivers for that source are responding to the transmitted information.

We can convert an isolated bus system into a bidirectional bus system using bus transceiver integrated circuits as shown in Fig. 7. Transceivers are built into the 2606 and 2111 (they don't have enough pins to do otherwise), and may be externally added to regular 2102 type memories. In the case of a TV typewriter, it's usually desirable to keep the display and its memory source (either its own or the memory of a microcomputer) connected together; this eliminates dropouts when the bus is going downstream. Thus, the display electronics is best placed between the memory read outputs and the bus transceiver.

Address lines can also share the same data bus as the input and output data lines, but this leads to extremely difficult timing, particularly in 8-bit systems. More often, the address bus will remain separate but will be able to accept address commands from several sources. These sources could include the TV typewriter's live scan timing, a cursor address for update during retrace, and an optional external dominant microcomputer control for rapid and wholesale screen changes.

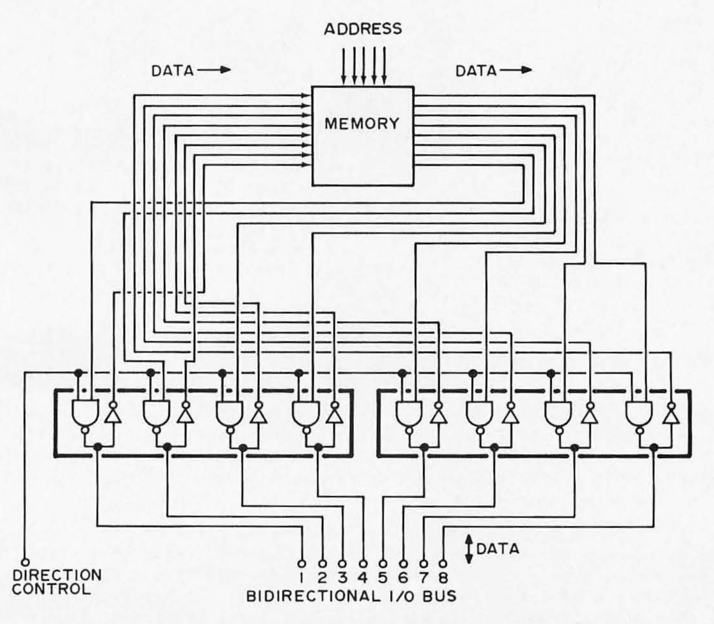


Fig. 7. Connecting a memory subsystem to a bidirectional input output bus is required in most microcomputer applications.

Computers Are RIDICULOUSLY SIMPLE!

Did you just get hooked? Has the first reaction of bewilderment and perplexity set in as you begin to explore the ins and outs of computing? Nat Wadsworth of SCELBI Computer Consulting – makers of an Intel 8008 based packaged microcomputer system – provides us with this article on fundamentals of computer operation. The article is written with the Intel 8008 in mind as an example of a typical computer, but the principles involved apply to nearly any microcomputer you can find on the market. The material of this article is taken from the first chapter of author Wadsworth's SCELBI-8H/B User's Manual, *one of the best documentation support packages among the various kit manufacturers.*

by
Nat Wadsworth
SCELBI Computer Consulting Inc.
1322 Rear, Boston Post Road
Milford CT 06460

There have been numerous examples put forth over the years to illustrate the basic scheme behind the operation of computers. The scheme is deceptively simple and incredibly powerful. The power comes from the speed with which the machines can perform the simple operations. The fundamental concept of the computer is that it is a machine that is capable of doing two fundamental operations at very high speed: First it is able to obtain a piece of information from a storage area and perform a function as directed by the information it obtains; and secondly, based on its current status, it is able to ascertain where to obtain the next piece of information that will give it further "directions." This fundamental concept is the key to the operation of all digital computers and while it is a simple concept, it can be built upon to arrive at all the complex operations computers of today can perform. How this is done is what this article is about.

One of the best analogies for describing a computer's basic operations is to consider a bank of boxes, similar to a bank of Post Office mail boxes. A piece of paper containing "directions" can be placed in each box. A person is directed to go to the bank of boxes, and after starting at a given place, to open each box, withdraw the piece of paper and follow the directions there-on. The boxes are labeled in an orderly fashion, and the person is also told that unless a piece of paper in a box directs otherwise, when the person is finished performing the task directed, they are to replace the paper in the box and proceed to open the next box. Note, however, that a piece of paper may give directions to alter the sequence in which the person is to open boxes.

Fig. 1 shows a picture of a set of such boxes. Each box is labeled for identification.

To present a view of a computer's operation, assume a person has been told to start at box A1 and to follow the directions contained on the pieces of paper in the boxes until a piece of paper containing the direction "stop" is found in one of the boxes. In this example the person finds the following "instructions":

In box A1 is the message: *"Take the mathematical value of 1 and write it down on a scratch pad."*

Since the "instruction" in box A1 only pertained to some function that the person was to perform, and did not direct the person to go to some specific box, then the person will simply go on to the next box in the row.

Box A2 contains the information:

"Add the number 2 to any value already present on your scratch pad."

The person will at this point perform an addition and have a total "accumulated" value on the pad of scratch paper. The accumulated value would be 3. Since there are no other directions in box A2, the operator would continue on to open box A3 which has the following message:

"Place any accumulated mathematical value you have on your scratch pad into box H8."

Thus the person would tear the current sheet off the "scratch pad" and place it – containing the value "3" – into box H8. Note, though, that while the person was directed to place the accumulated value on the scratch pad into box H8, the person was not directed to alter the sequence in which to obtain new "instructions" so the person would proceed to open box A4 which contains the directive:

"Take the mathematical value of 6 and place it on your scratch pad."

Going on to box A5 the person finds:

"Add 3 to the present value on your scratch pad."

This is obviously just a "data word." The operator adds the value 6 from the previous box to the number 3, noting the calculation on the scratch pad and proceeds to open box A6:

"Place any accumulated value you have on your scratch pad into box H7."

The person thus would put the value "9" on a piece of paper (from the scratch pad) into the designated box and proceed to open box A7:

"Get the value presently stored in box H8 and save the value on your scratch pad."

This is a simple operation and the person proceeds to open up box A8:

"Fetch the value in box H7. Subtract the value of your scratch pad from the value found in box H7. Leave the result on your scratch pad."

When the operator has performed this operation, the operator will have finished the "A" row and will then continue obtaining "instructions" by going to the "B" row and opening box B1 where more directions are found:

"If the present value on your scratch pad is not zero go to box B3."

At this time if the person checks the scratch pad it will be found that the value on the scratch pad is indeed non-zero as the last calculation performed on the scratch pad was to subtract the value in box H8 from the value in box H7. In this example that would be:

$$9 - 3 = 6$$

Therefore the directions in box B1 for this particular case will tell the operator to "jump over" box B2 and go to box B3. For the sake of completeness, however, box B2 does contain an instruction, for had the value on the scratch pad been zero the operator would not have "jumped over" box B2 and would have found the following message inside box B2:

"The values in box H7 and H8 are of equal value. STOP!"

However, for the values used in this example, the person would have "jumped" to box B3 where the following directive would be found:

"If the present value on your scratch pad is a "negative number" jump to box B5."

Since this is not currently the case the person will not "JUMP" to box B5, but will simply continue to open box B4 which contains:

"The value in box H7 is larger than the value in box H8. STOP!"

At this point the person has completed the "instruction sequence" for this example. It should be noted, however, that box B5 did contain the message:

"The value in box H7 is smaller than the value in box H8. STOP!"

This little example of a person opening up boxes and following the directions

The basic scheme behind the operation of computers is deceptively simple and incredibly powerful.

One of the best analogies for describing a computer's basic operations is to consider a bank of boxes, similar to a bank of Post Office mail boxes . . .

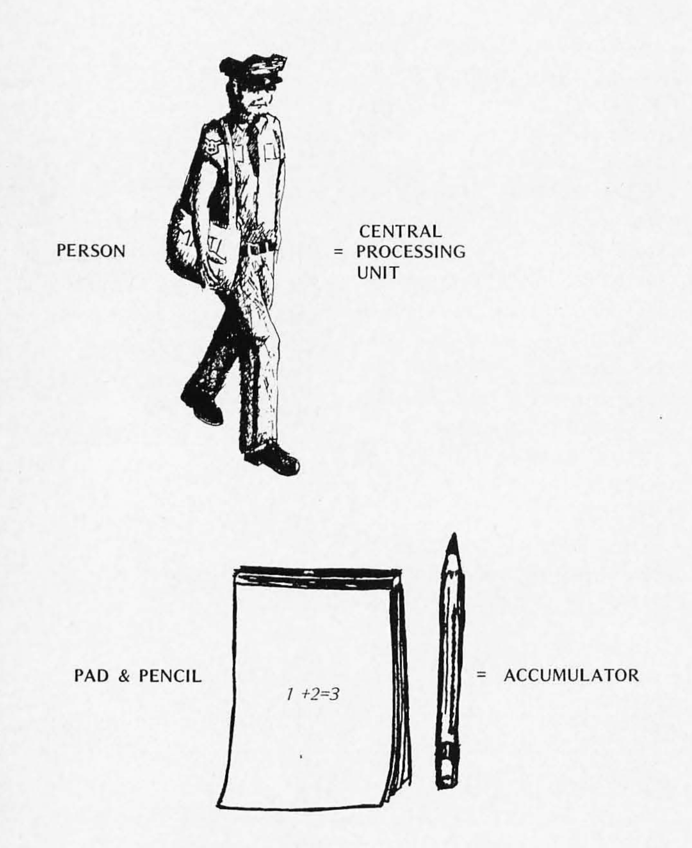
A1	A2	A3	A4	A5	A6	A7	A8
B1	B2	B3	B4	B5	B6	B7	B8
C1	C2	C3	C4	C5	C6	C7	C8
D1	D2	D3	D4	D5	D6	D7	D8
E1	E2	E3	E4	E5	E6	E7	E8
F1	F2	F3	F4	F5	F6	F7	F8
G1	G2	G3	G4	G5	G6	G7	G8
H1	H2	H3	H4	H5	H6	H7	H8

Fig. 1. A set of Post Office pigeon holes containing messages.

Fig. 2(a). The computer structure compared to the Post Office pigeon holes.

POST OFFICE BOXES = MEMORY

A1	A2	A3	A4	A5	A6	A7	A8
B1	B2	B3	B4	B5	B6	B7	B8
C1	C2	C3	C4	C5	C6	C7	C8
D1	D2	D3	D4	D5	D6	D7	D8
E1	E2	E3	E4	E5	E6	E7	E8
F1	F2	F3	F4	F5	F6	F7	F8
G1	G2	G3	G4	G5	G6	G7	G8
H1	H2	H3	H4	H5	H6	H7	H8



contained in each one is very similar to the concept used by a computer. Note that each "instruction" is very short and specific. Also note that the combination of all the instructions in the example will result in the person being directed to solve the problem:

Is 1 + X greater than, less than, or equal to: 6 + Y?

For the reader can note, if the "data words" contained in boxes A2 and A5 for the example were changed, the sequence of "instructions" would still result in the person being told to "STOP" at the box that contained the correct answer. The reader can verify this by simply assuming that different numbers than those used in the example are in boxes A2 and A5 and going through the instruction sequence until told to "STOP."

The example illustrates how a carefully planned set of directions, arranged such that they are performed in a precise sequence, can be used to solve a problem even though the "variables" (data) in the problem may vary. Such a set of "instructions" is often termed an "algorithm" by those in the computer field. The example solved a mathematical problem using the "algorithm," but the reader will find that "algorithms" can be devised to solve many problems on a computer that are not strictly mathematical!

Any person learning a new skill must of necessity learn the vocabulary of the field in order to proceed to any great extent. You might think that it would be easier if everything was written in plain everyday words, but the truth of the matter is that specialized vocabularies do serve several useful functions. For one thing, they can greatly shorten the time that it takes to communicate ideas or concepts. In today's fast-moving world, that is of significance in itself. In addition, the limitations of the English language often result in a given word having a special meaning when it is used in the context of a particular subject. One must know the new meaning when it is used in such a manner. Fortunately, much of the computer vocabulary is very logically named. This is probably due partly to the fact that computers are of necessity extremely dependent on logic, and hence many persons who helped create the field – and by that fact were rather logically oriented themselves – seem to have had the logical sense to have named many of the parts and systems of computers and computer programs, in a logical manner.

In the text which follows, two diagrams, Figs. 2(a) and 2(b), are used to demonstrate the analogy between the person taking "instructions" from a group of mail boxes and the basic operation of a real minicomputer.

Fig. 2(a) shows the Post Office boxes, a figure representation of a person who is able to "fetch" and return the "instructions" or "data" from and to the boxes, and a "scratch pad" on which the person can make temporary calculations when directed to do so.

In Fig. 2(b) are three interconnected boxes which form a "block diagram" of a computer. The uppermost portion of the "block diagram" is labeled the "memory." The middle portion is labeled the "central processor unit" or "CPU" for short. The lower part of the diagram depicts an "accumulator."

The correlation between the two pictures is extremely simple. The "Post Office boxes" correspond to the "memory" portion of a real computer. The "memory" is a storage place, a location where instructions and data can be stored for long lengths of time. The "memory" can be "accessed." "Instructions" and/or "data" can be taken out of memory, operated on, and replaced. New "data" can be put into the "memory." A "memory" that can be "read from" as well as "written into" is called a "read and write memory." A "read and write memory" is often referred to as a "RAM" as an abbreviation. Many times it is feasible to have a "memory" that is only "read from." A memory that is never "written into," but is only used to "read from," is termed a "read only memory" and is abbreviated as a "ROM." For the present discussion the term "memory" will refer to a "read and write memory" ("RAM").

The figure of a person in Fig. 2(a) corresponds to the central processor unit in Fig. 2(b). The central processor

unit in a computer is the section that "controls" the overall operation of the machine. The "CPU" can receive (fetch) "instructions" or "data" from the memory. It is able to "interpret" the "instructions" it fetches from the memory. It is also able to perform various types of mathematical operations. It can also "return" information to the memory – for instance make deposits of "data" into the memory. The "CPU" also contains control sections that enable it to sequentially "access" the "next" location in memory when it has finished performing an operation, or, if it is directed to do so, to "access" the memory at a specified location, or to "jump" to a new area in memory from which to continue fetching "instructions."

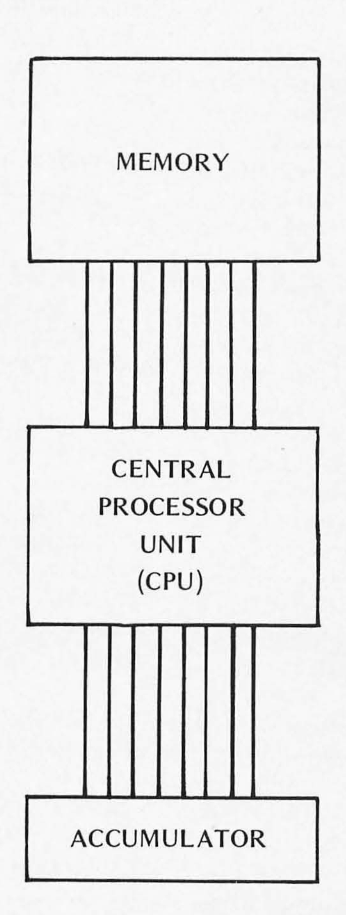
The pad of paper and pencil in Fig. 2(a) corresponds to the block titled "accumulator' in Fig. 2(b). The "accumulator" is a temporary "register" or "manipulating area" which is used by the CPU when it is performing operations such as adding two numbers. One number or piece of information can be temporarily held in it while the central processor unit goes on to obtain additional instructions or data from memory. It is an electronic "scratch pad" for the CPU.

The three fundamental units – the memory, central processor unit, and the accumulator – are at the heart of every digital computer system. Of course, there are other parts which will be added in and explained later, but these fundamental portions can be used to explain the basic operation of a digital computer which is the purpose of this article.

The reader should learn the names of the basic parts of the computer as they are presented. Note how easy it is to remember the portions that have been shown. The "remembering" element is a "memory." The portion that does the "work" or processing is simply termed the "central processor unit," and the part that is used to accumulate information temporarily is aptly called the "accumulator!"

The reader should now have a conceptual view of the concept behind a computer's operation and an understanding of the machine's most basic organization. It is simply a machine that can fetch information from a memory, interpret the information as an instruction or data, perform a very small operation, and continue on to determine the next operation that is to be performed. Each operation it is capable of doing is very tiny by itself, but when the many operations of a typical "program" are performed in sequence, the solutions to very complex problems can be obtained. It is important to remember that the computer can perform each little operation in just a few millionths of a second! Thus a program that might seem very large to a person – say one with many thousands of individual instructions – would only take a digital computer a few thousandths of a second to perform. The speed with which the computer can execute individual instructions is what gives the computer its seemingly fantastic capability.

Fig. 2(b). Block diagram of a computer's fundamental components.



It is now time to start delving into the actual physical manner in which a computer operates. How can a machine be constructed so that it is able to perform the processes of the central processor unit? While it will require a number of pages of text to explain the procedure, it is not nearly as difficult to understand as many people might suspect. The complexity of a computer when first viewed by a person is caused by the fact that it appears to consist of many hundreds of parts. It becomes much simpler when one understands that the hundreds of parts are really made up from a few dozen similar parts and they are carefully organized into just a few major operating portions. The reader is already familiar with the most fundamental portions.

As fantastic as it may sound at first, a digital computer can be thought of as really nothing more than a highly organized collection of "on or off" switches! Yes, computers are constructed from electronic devices that can only assume one of two possible states! The electronic switches can be constructed in a variety of ways. For instance, the switch can be made so that the voltage at a given point is either high or low, or current through a device is either flowing or not flowing, or flowing in one direction, and then the other direction. But, regardless of how the electronic switch is constructed, its status can always be represented as being either "on" or "off." This "on" or "off" status can be mathematically symbolized most suitably by a mathematical system based on "binary" notation.

Some people tend to think that computers are very difficult to understand because they have heard of "strange" types of mathematics that are often

As fantastic as it may sound at first, a digital computer can be thought of as really nothing more than a highly organized collection of "on or off" switches!

referred to in conjunction with computers. In actuality much of the mathematics that are dealt with in computer technology are much easier to understand and deal with than the decimal system that the average person is familiar with. In the decimal numbering system a person must learn 10 different symbols, and in order to manipulate those symbols, they must memorize a lot of information. For instance, look at how students are taught to multiply. The learning process actually involves the student having to memorize a rather large number of facts. Because of the way it is typically taught, most students never realize how much work they have to go through just to learn the multiplication tables! The teacher does not stand up and say, "OK, now you are going to memorize about 100 facts." Instead, over a period of a few weeks or so, the student is made to memorize the 100 or so facts — a few at a time. The student must learn the value of each digit multiplied by all the other digits in the decimal numbering system. The decimal numbering system is far more complicated for the beginner than learning the binary numbering system, and the binary numbering system is the one utilized by computers at their most basic functioning level. The reason the computer uses the binary system is because it is the simplest system around and hence the easiest one with which to construct a computing machine!

Readers know the word "binary" indicates "two." Computers are built up of electronic switches that can only have two possible states. The switches are binary devices. The status of the switches can be represented mathematically utilizing the "binary" numbering system. The binary numbering system only has two digits in it! They are zero (0) and one (1). A switch can thus be mathematically symbolized, for instance, by a zero when it is "off" and a one when it is "on." The opposite relationship could also be established, a one could be used to represent a switch being "off" and a zero used to represent a switch as "on." It would make no difference mathematically which convention was used as long as one was consistent. For the purposes of the present discussion, the reader can assume that the first convention (switch off = 0, switch on = 1) will be used.

It should be immediately apparent that working with a numbering system based on only two integers will be a lot easier than working with one having 10 integer symbols. In fact, most problems for people learning the binary system come about because they tend to forget how simple it is, and they tend to keep going towards a decimal solution out of habit when they are working with the binary system. For instance, when one starts to add binary numbers, as soon as the value "1" is exceeded, a "carry" to the next column must be made. The value of the addition of "1 + 1" in the binary system is: 10. It is not 2! There is no such integer as "2" in the binary numbering system. However, when a person who has worked with the decimal system for years first starts working with the binary system, old decimal habits tend to get in the way. The reader will have to beware!

To formally introduce the binary mathematical system one can start by stating that it uses two integers, zero (0) and one (1), and no others. A binary number has a value determined by the value of the integers that make up the number, and the position of the digits.

In the decimal numbering system, the reader is familiar with the location of a digit having a "weighted" value as follows: A three digit number has a value determined by the unit value of the digit in the right-most column plus the value of the digit to the left of it multiplied by 10, plus the value of the third digit multiplied by one hundred as illustrated in the following example:

THE DECIMAL NUMBER
345 IS EQUAL TO:

5 UNITS = 5
PLUS(+) 4 TIMES 10 = 40
PLUS (+) 3 TIMES 100 = 300

In other words, after the right-most column (which has the value of the digit), each column to the left is given a weighting factor which increases as a power of the total number of digits utilized by the numbering system. Note that in the above example the 4 representing 40 units is equal to 4 times the number of integer symbols in the decimal system (10) because it is located in the second column from the right. The number 3 representing 300 units is equal to 3 times the number of integer symbols in the decimal system squared because it is located in the third column from the right. This relationship of the weighted value of the digits based on their position can be described in mathematical shorthand as follows:

If the number of different integer symbols in the numbering system is U (for the decimal system U=10)

and the column whose weighted value is to be determined is column number M (starting with the right-most column and counting to the left)

and any digit is represented by the symbol X

then the weighted value of a digit in column M is expressed as:

X times U raised to the power (M-1) or $XU^{(M-1)}$

The reader can easily verify that the above formula applies to the decimal numbering system. However, the above formula is a general formula that can be used to determine the weighted positional value of any numbering system. It will be used to determine the weighted positional values of numbers in the binary numbering system.

In the binary numbering system there are just two different integer symbols (0

and 1). Thus U in the above formula is equal to 2. For illustrative purposes assume the following binary number is to be analyzed:

101

and it is desired to determine its value in terms of decimal numbers. (Remember its binary value is just 1 0 1). Using the above formula for the digit in the right-most column: M is equal to 1, thus (M-1) is equal to 0, and with X = 1:

$$\text{Weighted Value} = X \,.\, U^{(M-1)} = 1 \,.\, 2^0 = 1$$

(Remember that any number raised to the zero power is equal to 1.) Going on to the next digit it can be seen that the weighted value is simply 0! Finally, the digit in the third column from the right has the weighted value because of its position:

$$\text{Weighted Value} = X \,.\, U^{(M-1)} = 1 \,.\, 2^{(3-1)} = 2^2 = 4$$

Then, by adding up the sum of the weighted values (similar to that done for the decimal example earlier) one can see that the decimal equivalent of 1 0 1 binary is 5:

THE BINARY NUMBER 101

IS EQUAL TO:

1 UNITS	=	1
+ 0 TIMES 2	=	0
+ 1 TIMES 4	=	4

and thus 1 0 1 in the binary numbering system is the same as 5 in the decimal numbering system.

There will be more to learn about the binary numbering system. However, the brief information given will be enough to continue on with the discussion that this section is primarily concerned with — the basic operation of a computer. Since the reader is now aware that a computer is composed of numerous electronic switches and knows that one can use a mathematical shorthand to represent the status of the switches (whether they are "on" or "off"), and is also aware of the fundamental concept behind a computer's operation, it is now possible to proceed to show how electronic switches can be arranged to build a functional computer. That is, how the electronic switches can be arranged and interconnected in a fashion that will allow a machine to "fetch" a piece of information from a "memory" section, decode the information so as to determine an "instruction," and also determine where to obtain the next instruction or additional "data."

To begin this part of the discussion it will be beneficial for the reader to picture a group of cells (similar to the Post Office boxes shown earlier) arranged in orderly rows as shown in Fig. 3. This time, instead of each cell holding a complete instruction, it can be understood that each cell only represents part of an instruction and that it takes a whole row of cells to make up an instruction. Furthermore, each cell may only contain the mathematical symbol for a one (1) or a zero (0) — or, in other words, its contents represent the status of an electronic switch!

At this time a few more computer technology definitions will be illustrated. In Fig. 3, each box containing a binary 1 or 0 represents what is called a "bit" of information. While each cell may only contain one piece of information at a time, a cell can actually represent one of two possible states of information. This is because the cell can be in two possible states — it either contains a zero or a one. If one starts assigning positional values to the cells in a row, it can be seen that the total number of possible states in one row will increase rapidly. For instance, two cells in a row can represent up to four states of information. This is because two cells side-by-side, containing either a 0 or 1 in each cell can have one of the following four states at a particular moment in time: 1

The decimal numbering system is far more complicated for the beginner than learning the binary numbering system, and the binary numbering system is the one utilized by computers at their most basic functioning level.

WORD #1	1	0	1	0	1	0	1	0
WORD #2	0	1	0	1	0	1	0	1
WORD #3	1	1	0	0	1	1	0	0
WORD #4	0	0	1	1	0	0	1	1
WORD #5	1	1	1	1	0	0	0	0
WORD #6	0	0	0	0	1	1	1	1
WORD #7	1	1	1	1	1	1	1	1
WORD #8	0	0	0	0	0	0	0	0

Fig. 3. An array of electronic cells, 8 bits per cell.

The combination of the eight cells can be filled with zeros and ones in 256 different patterns.

0, 0 1, 1 1, or 0 0. Three cells in a row can represent up to eight states of information as the possible states of three cells side-by-side are: 0 0 0, 0 0 1, 0 1 0, 0 1 1, 1 0 0, 1 0 1, 1 1 0, 1 1 1. In fact, when each cell can represent a binary number, the total number of states of information that a row of "N" cells can represent is: 2 to the Nth power, 2^n. Thus, a row of eight binary cells can represent 2 to the eighth (256) states of information! That is, the combination of the eight cells can be filled with zeros and ones in 256 different patterns!

A group (row) of cells in a computer's memory is often referred to as a "word." A "word" in a computer's memory is a fixed size group of cells that are "accessed" or manipulated during one operational cycle of the central processing unit (CPU). The CPU will effectively handle all the cells in a "word" in memory simultaneously whenever it processes information in the memory. Digital computers can have varying "word lengths" depending on how they are engineered. Many microcomputers have a memory word size consisting of eight cells. The number of cells in a word, and the number of words in a computer's memory have a lot to do with the machine's overall capability. In the typical microcomputer system, the memory is available in modules – groups of words which can be plugged into a common set of wires in the system. With current LSI technology, a typical module of moderate price has 1024 bytes in an 8-bit computer system. With the 8008 oriented design serving as the basis for this article, one could potentially plug in 16 such modules for a total of 16,384 bytes or 131,072 bits. Thus, a large amount of information can be "stored" in the computer's memory at any one time.

The astute reader may have already figured out a very special reason for grouping cells into "words" in memory. It was pointed out earlier that a row of eight cells could represent up to 256 different patterns. Now, if each possible pattern could be "decoded" by electronic means so that a particular pattern could specify a precise "instruction" for the central processor unit, then a large group of "instructions" would be available for use by the machine. That is exactly the concept used in a digital computer. Patterns of ones and zeros organized into a computer "word" are stored in memory. The CPU is able to examine a word in memory and decode the pattern contained therein to determine the precise operation that it is to perform. Most microcomputers do not decode every one of the possible 256 patterns that can be held in a row of eight cells as an instruction. They have an "instruction set" of over 100 "instructions" which are represented by different patterns of ones and zeros in an eight cell memory "word." Each pattern that represents an "instruction" can be decoded by the CPU and will cause the CPU to perform a specific function. Details of all the functions a computer can perform are usually found in the manufacturer's documentation.

There is another ingredient necessary for making the machine "automatic" in operation. That is that the CPU must "know" where to obtain the next "instruction" in memory after it completes an operation. That function is greatly aided by having the memory cells grouped as "words." The reader should note that in Fig. 3 each group of cells representing a word was labeled as: "word #1," "word #2," etc. There is a special portion of the central processor unit that is used to control where the next word containing an instruction in memory is located. This special part is commonly referred to as the "program counter." One reason it was given the name "program counter" is because most of the time all it does is count! It counts memory words! Each word in memory is considered to have an "address." In Fig. 3 each word was given an "address" by simply designating each word with a number. Word #1 has an "address" of 1. Word #2 has an address of 2, etc. The "program counter" portion of the CPU keeps tabs on where the CPU should obtain the next instruction by maintaining an "address" of the word in memory that is to be processed! About 90% of the time all the program counter does is "increment" the value it has each time the CPU finishes doing an operation. Thus, if the computer were to start executing a simple program that began by its performing the instruction contained in "word #1" in memory – the very process of having the machine start the program at that location in memory would cause the program counter to assume a value of 1. As soon as the CPU had performed the function the "program counter" would increment its value to 2. The CPU would then look at the program counter and see that its next instruction was located in word #2 in memory. When the instruction in word #2 has been processed the "program counter" would increment its value to 3. This process might continue uninterrupted until the CPU found an instruction that told it to "STOP."

A sharp reader might be starting to ask, "Why have a program counter if each instruction follows the next?" The answer is simply that the availability of a "program counter" gives the freedom of not having to always take the instruction at the next "address" in memory. This is because the contents of the "program counter" can be changed when the CPU detects an "instruction" that directs it to do so! This enables the computer to be able to "jump" around to different sections in memory, and as will become apparent later, greatly increases the capability of the machine.

Fig. 4. The program counter of an 8008 based machine.

13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

The "program counter" is actually just a group of cells in the CPU that may contain either a binary zero or one. The binary value in the row of cells that constitute the program counter determines the "address" of a word in memory. Since the number of words in memory can be very large, and since the program counter must be capable of holding the address of any possible location in memory, the number of cells in a row in the program counter is larger than the number of cells in a word in memory. In an 8008 oriented computer design, for example, the number of cells in the program counter is 14. Since 2 to the 14th power is 16,384, the program counter can present up to 16,384 different patterns. Each pattern can be used to represent the "address" of a word in memory. Fig. 4 illustrates what the contents of the program counter would look like when it contained the address for a specific word in memory. The address the example displays is "address 0" which can be considered the first word in memory. The reader should note that an address of zero can actually represent a word in memory!

Earlier it was stated that some "instructions" can actually change the value of the program counter and thus allow a program to "jump" to different sections in memory. However, the reader now knows that a word in memory only contains eight cells, and yet the program counter of an 8008 based computer contains 14 cells. In order to change the entire contents of the program counter (by bringing in words from memory), it is necessary to use more than one memory word! This can be done if the program counter is considered to actually be two groups of cells connected together. One group contains eight cells, and the other six. In order to change the contents of the entire program counter, one whole eight cell word could be read from a memory location and placed in the right-hand group of eight cells of the program counter. Then another eight cell word could be read from memory. Since only six more cells are needed to finish filling the program counter, the information in two of the eight cells from the second word brought in from memory could be "discarded." If the information in the two left most cells of the word in memory were thrown away then the remaining six cells would contain information that could be placed in the six unfilled locations in the program counter. Most of the common 8-bit micro-computers use a similar scheme of breaking an address into two pieces when the program counter is loaded in a jump instruction.

In order to make it easier for a person working with the machine to remember "addresses" of words in memory, a concept referred to by computer technologists as "paging" is utilized. "Paging" is the arbitrary assignment of "blocks" of memory words into sections that are referred to figuratively as "pages." The reader should realize that the actual physical memory unit consists of all the words in memory – with each word assigned a numerical address that the machine utilizes. As far as the machine is concerned, the words in memory are assigned consecutive addresses from word #0 on up to the highest word # contained in the memory. However, people using computers have found it easier to work with addresses by arbitrarily grouping "blocks" of words into pages. For example in the Intel 8008 "pages" are considered to be "blocks" of 256 memory words. The first memory word address in an 8008 system is at address zero (0). Programmers could refer to this word as word #0 on page #0. The 256th word in memory as far as the computer is concerned has an address of 255. (Note: Since the address of 0 is actually assigned for the first physical word in memory, all succeeding words have an address that is one less than the physical quantity!) A programmer could refer to this word as word #255 on page #0. The 257th word in memory has an absolute address of 256 ("n"th word minus one since location 0 contains a memory word) as far as the machine is concerned, but a programmer could refer to that word location as being on page #1 at location 0! Similarly, the 513th word in memory, when the paging concept is used, becomes word #0 on page #2 for a programmer – but it is just 512 as far as the machine is concerned. Paging at multiples of 256 is a convenient tool when dealing with any 8-bit micro-computer.

The reader might have noted a nice coincidence in regards to the assignment of "paging" in 8-bit computers. Each "page" refers to a

There is a special portion of the central processor unit (CPU) that is used to control where the next word containing an instruction in memory is located – the "program counter." Most of the time all it does is count!

Fig. 5. The program counter with address 511 represented in binary notation.

13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1	1	1	1	1	1	1

"block" of memory words that contains 256 locations (0 to 255). The reader will recall that that is exactly the number of different patterns that can be specified by a group of eight binary cells, and there are eight binary cells in a memory "word." The relationship is more than coincidental! Note that now one has devised a convenient way for a person to be able to think of memory addresses and at the same time be able to specify a new address to the program counter that will still result in it containing an "absolute" address that the machine can use. For instance, if it was desired to change the contents of the 14 cell program counter from an absolute address of word #0, say to word #511, the following procedure could be used: The programmer would first specify an instruction that the CPU would decode as meaning "change the value in the program counter." (Such an instruction might be a "jump" instruction in the instruction set.) Following that instruction would be a word that held the desired value of the "low order address" or word # within a "page." Since a memory word only has eight cells, since eight cells can only represent 256 different patterns, and since one of the patterns is equivalent to a value of zero, then the largest number the eight cells can represent is 255. However, this is the largest word # that is contained on a page. This value can be placed in the right-most eight cells of the program counter. Now it is necessary to complete the address by getting the contents of another word from memory. Thus, immediately following the word that contained the "low address" would be another word that contained the "page #" of the address that the program counter was to contain. In this case the page number would be 1. When this value is placed in the left six cells of the program counter the program counter would contain the pattern in Fig. 5.

If desired, the reader can verify by using the formula presented previously for determining the decimal value of a binary number, that the pattern presented in Fig. 5 corresponds to 511, and thus, by using the "page #" and "word # on the page," each of which will fit in an eight cell memory word, a method has been demonstrated that will result in the program counter being set to an absolute address for a word in memory. Fig. 6 provides some examples as a summary.

By now the reader should have a pretty good understanding of the concepts regarding the organization of memory into electrical cells which can be in one of two possible states, the grouping of these cells into "words" which can hold patterns which the CPU can recognize as specifying particular operations, and the operation of a "program counter" which is able to hold the "address" of a word in memory from which the CPU is to obtain an instruction.

It is now time to discuss the operation of the "scratch pad" area for a computer – the accumulator (and some

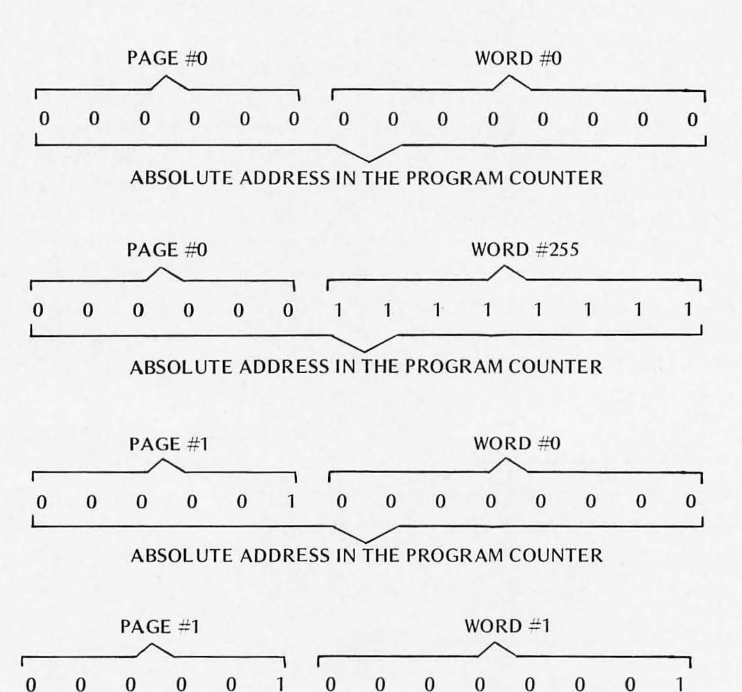


Fig. 6. Examples of addresses in an 8008 based system.

additional "manipulating registers" in the typical 8008 based computer).

As was pointed out earlier, there is a section of a computer that is used to perform calculations in and which can hold information while the CPU is in the process of "fetching" another instruction from the memory. The portion was termed an "accumulator" because it could "accumulate" information obtained from the CPU performing a series of instructions until such time as the CPU was directed to transfer the information elsewhere (or discard it). The accumulator is also considered to be the primary "mathematical" center for computer operations for it is the place where additions, subtractions, and various other mathematically oriented operations (such as Boolean algebra) are generally performed under program control.

The concept of an "accumulator" is not difficult to understand and its physical structure can be readily explained. The actual control of an accumulator by the CPU can be quite complex, but these complex electronic manipulations do not have to be understood by the computer user. It is only necessary to know the "end results" of the various operations that can be performed within an accumulator.

The accumulator in an Intel 8008 based machine can be considered as a group of eight "memory cells" similar to a "word" in memory except that the information in the cells can be manipulated in many ways that are not directly possible in a word in memory.

Fig. 7 shows a collection of eight binary cells containing ones and zeros to represent an accumulator. The cells are numbered from left to right starting with "B7" down to "B0." The designations refer to "bit positions" within the accumulator. Note that the right-most cell is designated B0 and the eighth cell (left-most cell) is designated B7. The reader should become thoroughly familiar with the concept of assigning the reference of "zero" to the right-most bit position in a row of cells (similar to the concept of assigning a reference of zero to the first address of a word on a page in memory) as the convention is frequently used by computer technologists. The convention can be confusing for the beginner who fails to remember that the physical quantity is one more than the reference designation. The convention of labeling the first physical position as zero makes much more sense once the reader learns to think in terms of the binary numbering system and thoroughly realizes that the "zero" referred to so frequently in computer work when discussing actual operations actually represents a physical state (the status of an electronic switch) and does not necessarily imply the mathematical notion of "nothing." The concept of assigning a bit designation to the positions of the cells within the accumulator will allow the reader to follow explanations of various accumulator operations.

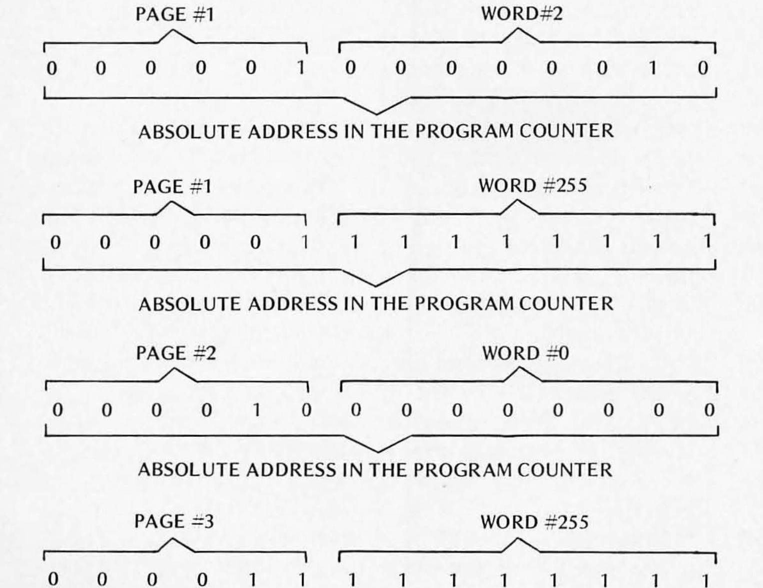
Fig. 7. The accumulator, pictured with binary 10101010 (decimal value 160) in its 8 bits.

B7	B6	B5	B4	B3	B2	B1	B0
1	0	1	0	1	0	1	0

The accumulator simply holds a number–it adds and subtracts–and "rotates" its contents.

One of the most fundamental and most often used operations of an accumulator is for it to simply hold a number while the CPU obtains a second operator. In an 8008 type of machine the accumulator can be "loaded" with a value obtained from a location in memory or one of the "partial accumulators." It can then hold this value until it is time to perform some other operation with the accumulator. (It will become apparent later that the accumulator of an 8008 can also receive information from external devices.)

Perhaps the second most often used operation of an accumulator is to have it perform mathematical operations such as addition or subtraction with the value it contains at the time the function is performed and the contents of a memory location or one of the "partial accumulators." Thus if the accumulator contained



259th PHYSICAL WORD IN MEMORY HAS AN ABSOLUTE ADDRESS OF: 258

512th PHYSICAL WORD IN MEMORY HAS AN ABSOLUTE ADDRESS OF: 511

513th PHYSICAL WORD IN MEMORY HAS AN ABSOLUTE ADDRESS OF: 512

1024th PHYSICAL WORD IN MEMORY HAS AN ABSOLUTE ADDRESS OF: 1023

Fig. 8. Adding the content of a memory word to the accumulator.

B7	B6	B5	B4	B3	B2	B1	B0	
0	0	0	0	0	1	0	1	ORIGINAL CONTENTS OF THE ACCUMULATOR
0	0	0	0	0	0	1	1	CONTENTS OF THE SPECIFIED WORD IN MEMORY
0	0	0	0	1	0	0	0	FINAL RESULTS AFTER THE ADDITION IN THE ACCUMULATOR

the binary equivalent of the decimal number 5, and an instruction to add the contents of a specific memory location which contained the binary equivalent of the decimal number 3 was encountered, the accumulator would end up with the value of 8 in binary form as shown in Fig. 8.

Perhaps the next most frequently used group of operations for the accumulator is for it to perform ''Boolean'' mathematical operations between itself and/or other ''partial accumulators'' or words in memory. These operations in the typical microcomputer include the logical "and," "or," and "exclusive or" operations.

Another important capability of the accumulator is its ability to "rotate" its contents. In an 8008, as in many micros, the contents of the accumulator can be rotated either to the right or left. This capability has many useful functions, and is one method by which mathematical multiplication or division can be performed. Fig. 9 illustrates the concept of "rotating" the contents of the accumulator.

The astute reader may notice that the accumulator rotate capability also enables the accumulator to emulate a "shift register" which can be a valuable function in many practical applications of the computer.

The accumulator serves another extremely powerful function. When certain operations are performed with the accumulator the computer is capable of examining the results and will then "set" or "clear" a special group of "flags." Other instructions can then test the status of the special ''flags'' and perform operations based on the particular setting(s) of the "flags." In this manner the machine is capable of ''modifying'' its behavior when it performs operations depending on the results it obtains at the time the operation is performed!

In an 8008 based computer, there are four special flags which are manipulated by the results of operations with the accumulator (and in several special cases by operations with "partial accumulators"). These four flags are described in detail below. Other micros have similar condition flags.

The "carry flag" can be considered as a one bit (cell) extension of the accumulator register. This flag is changed if the contents of the accumulator should ''overflow'' during an addition operation (or ''underflow'' during a subtraction operation). Also, the "carry bit" can be utilized as an extension of the accumulator for certain types of "rotate" commands.

The "sign flag" is set to a logic state of "1" when the most significant bit (MSB) of the accumulator (or partial accumulator) is a "1" after certain types of instructions have been performed. The name of this flag derives from the concept of using two's complement arithmetic in a register where the MSB is used to designate the sign of the number in the remaining bit positions of the register – conventionally, a "1" in the MSB designates the number as a "negative" number. If the MSB of the accumulator (or partial accumulator) is "0" after certain operations, then the "sign flag" is zero (indicating that the number in the register is a positive number by two's complement convention).

The "zero flag" is set to a logic state of "1" if all the bits in the accumulator (or partial accumulator) are set to zero after certain types of operations have been executed. It is set to "0" if any one of the bits is a logic one after these same operations. Thus the "zero flag" can be utilized to determine when the value in a particular register is zero.

The "parity flag" is set to a "1" after certain types of operations with the accumulator (or partial accumulators) when the number of bits in the register that are a logic one is an even value (without regard to the positions of the bits). The "parity flag" is set to "0" after these same operations if the number of bits in the register that are a logic one is an odd value (1, 3, 5 or 7). The "parity flag" can be especially valuable when data from external devices is being received by the computer to test for certain types of "transmission errors" on the information being received.

In its simplest form, a group of switches can be used as an input device and a group of lamps as an output device!

In addition to the full accumulator previously discussed there are six other 8 bit registers in the Intel 8008 computer referred to as ''partial accumulators'' because they are capable of performing two special functions normally associated with an accumulator (in addition to simply serving as temporary storage registers). The full accumulator will often be abbreviated in this manual as "ACC" or "register A." The six ''partial accumulators'' will be referred to as "registers B, C, D, E, H and L."

Registers B, C, D, E, H and L of an 8008 are all capable, upon being directed to do so by a specific instruction, of either incrementing or decrementing their contents by one. This capability allows them to be used as "counters" and "pointers" which are often of tremendous value in computer programs. What makes them especially valuable in 8008 architecture is that when their contents are incremented or decremented the immediate results of that register will affect the status of the "zero," "sign," and "parity" flags discussed above. Thus it is possible for the particular contents of these registers to affect the operation of the computer during the course of a programs operation and they can be used to "guide" or modify a sequence of operations based on conditions found at the actual time a program is executed.

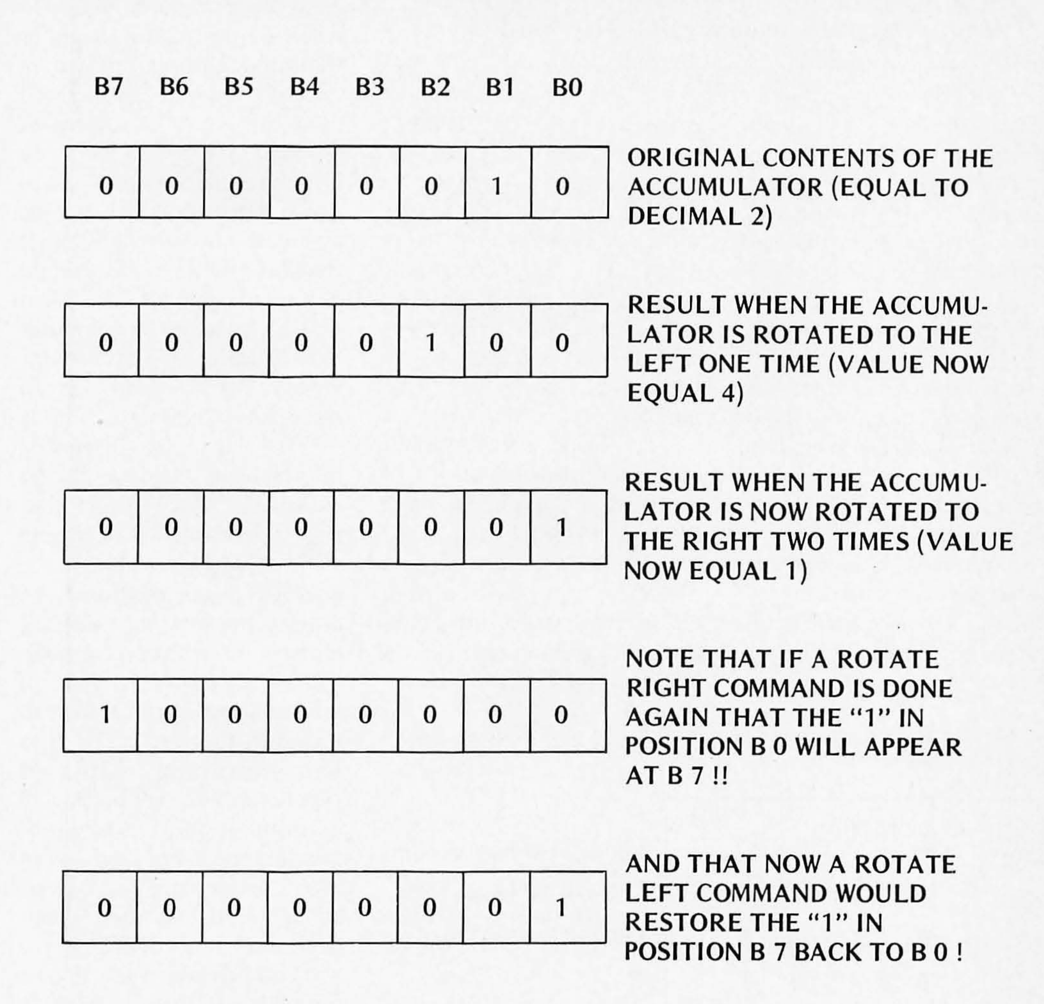


Fig. 9. Rotating the content of the accumulator.

It should be noted that registers B, C, D, E, H and L are capable of being incremented and decremented – but the full accumulator – register A – cannot perform those two functions in the same manner. (The full accumulator can be incremented or decremented by any value by simply adding or subtracting the desired value. There is not, however, a simple increment or decrement by one instruction for use with the full accumulator of an 8008!)

Two of the partial accumulators, registers H and L, serve an additional purpose in the 8008 computer CPU. These two registers can be used to directly "point" to a specific word in memory so that the computer may obtain or deposit information in a different part of memory than that in which a program is actually being executed. The reader should recall that a special part of the central processor unit (CPU) termed the program counter is used to tell the computer where to obtain the next instruction while executing a program. The program counter was effectively a "double word length" register that could hold the value of any possible address in memory. The program counter is always used to tell the machine where to obtain the next instruction. However, it is often desirable to have the machine obtain some information – such as a "data word" – from a location in memory that is not connected with where the next instruction to be performed is located. This can be accomplished by simply loading "register H" with the "high address" (page) portion of an address in memory, then loading "register L" with the "low address" portion of an address in memory, and then utilizing one of a class of commands that will direct the CPU to fetch information from or deposit information into the location in memory that is specified ("pointed to") by the "H" and "L" register contents. This information flow can be from/to the location specified in memory and any of the CPU registers.

At this time it would be beneficial for the reader to study Fig. 10. Fig. 10 is an expanded block diagram of Fig. 2(b) and shows the units of the computer which have been presented in the previous several pages.

Until now no mention has

It has been said that the computer is the most versatile machine in existence and that its applications are limited only by man's ability to develop programs that direct the operation of the machine.

The computer's great versatility comes about because the machine is capable of executing a large group of instructions in an essentially limitless series of combinations.

been made of how information is put into or received from a computer. Naturally, this is a very vital part of a computer because the machine would be rather useless if people could not put information into the machine upon which calculations or processing could be done, and receive information back from the machine when the operations(s) had been performed!

Communications between the computer and external devices — whether those devices be simple switches, or transducers, or teletype machines, or cathode-ray-tube display units, or keyboards, or "mag-tape" and "disk" systems — or whatever, are commonly referred to as input/output operations and are collectively referred to in abbreviated form as "I/O" transfers.

In the Intel 8008 computer designs all "I/O" transfers are typically made between external "I/O ports" (which connect to external devices via appropriate electronic connections) and the full accumulator in the computer. This I/O structure means that a whole group of devices can be simultaneously hooked up to the computer and the computer used to receive information from or transmit information to a variety of devices as directed by a "program." A special set of commands is used to instruct the computer as to which "I/O port" is to be operated at any particular instant. With appropriate programming it is then possible to have the computer "communicate" with a large variety of devices in an essentially "automatic" mode — for instance receiving information from a digital multimeter at specified times, then possibly performing some averaging calculations, and then outputting results to a teletype machine without human intervention. Or, in other applications — information from a human operator can be typed into the machine using a typewriter-like keyboard. In its simplest form, a group of switches can be used as an input device and a group of lamps used as an output device for the computer!

However, a more sophisticated system used in many applications would be to use a teletype machine or a combination of a keyboard and a cathode-ray-tube (CRT) display attached to input and output ports to serve as the primary means of I/O. A person can thus type information on the keyboard which will pass it into the computer, and the computer can display the results of its operations on the CRT display (which can, incidentally, be made from an ordinary oscilloscope and a special CRT interface unit such as that described in Jim Hogenson's article in BYTE #2).

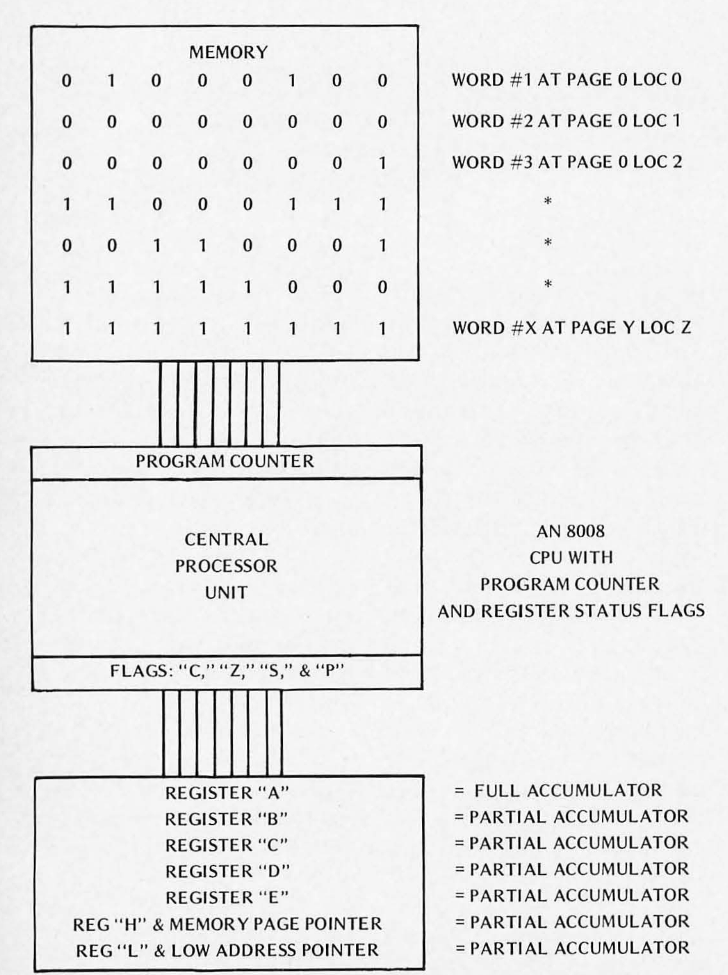
Perhaps the most wonderful and exciting aspect about a digital computer is its tremendous versatility. It has been said that the computer is the most versatile machine in existence and that its applications are limited only by man's ability to develop programs that direct the operation of the machine. It is undoubtedly one of the best machines for allowing man to exercise and test his creative powers through the development of programs that direct the machine to perform complex operations that can not only control other machines, or perform calculations many times faster than humanly possible, but because it can be used to "simulate" or "model" other systems that it might be impractical to build for purely experimental purposes. Thus man can create a "model" in a computer program and actually "play" with the synthetic model without actually building the physical device!

The computer's great versatility comes about because the machine is capable of executing a large group of instructions in an essentially limitless series of combinations — these series of instructions are stored in the memory bank(s) of the computer — and a new series of instructions can be placed in the memory bank(s) whenever desired. In fact, the memory bank(s) can often hold several completely unrelated "programs" in different sections and thus one can have a machine that performs totally unrelated tasks simply by pushing a few buttons and thereby directing the machine to start executing a new program in a different section of memory!

The digital computer is capable of providing services to people from all walks of life! A person need only choose (or develop) programs and connect external instruments that will provide the capabilities desired.

For instance, a scientist might put a mathematical calculator program into the

Fig. 10. The block diagram of Fig. 2(b) filled in with the designations for an Intel 8008 computer.



computer's memory and use the computer as a sophisticated electronic calculator by using a calculator-type keyboard as an input device and a CRT display as an output device on which to receive the answers to complex mathematical calculations which the computer performs. After using the computer as a calculator for a period of time, the scientist might decide to utilize the same computer to automatically record data from instruments during an experiment. By simply putting a different program in the computer's memory and plugging some peripheral measuring instruments into the computer's I/O ports, the scientist could have the computer periodically make measurements while he went out to lunch and save the results in its memory. After lunch the scientist could have the computer tabulate and present the data obtained from the experiment in compact form. Then, by merely putting a different program in the memory, the scientist could have the computer help him set up and arrange a "reference file" all sorted into alphabetical order or any manner that would enable him to use the computer to extract information far faster than a manually operated "paper file card" system.

The development of computer programs can be an extremely creative, exciting and personally rewarding pastime and offers essentially limitless ways to exercise one's creative capabilities . . . .

So the computer can be a valuable tool for a scientist; but, the same machine with a different program in its memory (and possibly different peripheral devices) could be used to control a complex manufacturing operation such as a plastic injection molding machine. In such a case I/O units that coupled to transducers on the injection molding machine might be used to relay information to the computer on a variety of parameters such as temperature of the plastic in the feed barrel, amount of feed material in the hopper and injection barrel, available pressure to the mold jaws and feed barrel, vacancy or filled status of the mold and other useful parameters. The computer could be programmed to analyze this information and send back signals to control the operation of heaters, pressure valves, the feed rate of raw materials, when to inject plastic into the mold, when to empty the mold, and other operations to enable the plastic injection system to operate in an essentially automatic mode.

Or, a businessman could use the same computer connected to an electric typewriter, with a suitable program in memory, to compose, edit and then type out "personalized form letters" by directing the computer to insert paragraphs from a "bank of standard paragraphs" so as to form a personalized customer answering system that would handle routine inquiries in a fraction of the time (and cost) that it would take a secretary to prepare such letters. Or, the businessman might utilize the computer to help him control his inventory, or speed up his accounting operations.

However, a computer that costs as little as the typical micro system does not have to be restricted to a business or scientific environment. The computer that can do all the types of tasks mentioned above can also be used to have fun with, or to perform valuable services, to private individuals.

The computer can be used as a sophisticated electronic calculator by almost anyone. It can be used to compose letters (using an editor program) by virtually anyone. Programs that sort data alphabetically or in various other categories can be of valuable service to people in many applications. The computer can be used to monitor and control many household items, serve as a security monitoring system, be connected to devices that will dial telephones, and do thousands of other tasks.

The electronic hobbyist can be kept occupied for years with a digital computer. For instance, one can build a little test instrument that plugs into a few I/O ports on the computer, then load programs into memory that will direct the computer to automatically test electronic components (such as complex TTL integrated circuits) in a fraction of a second! (Businesses can do this too!)

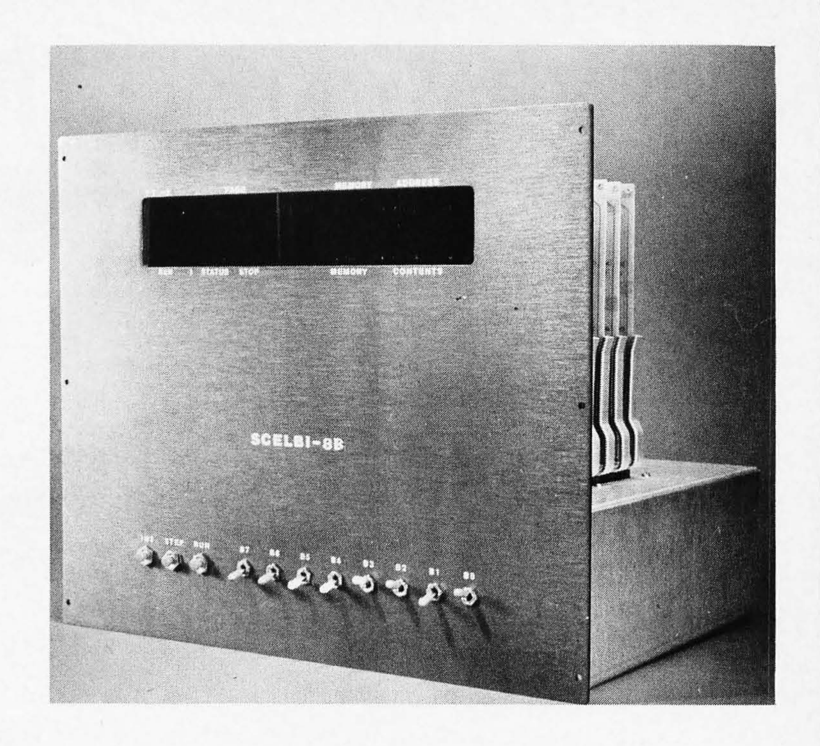
Or a ham radio operator can put a program into memory that will enable the computer to receive messages typed in from a keyboard, convert the messages to Morse code, and then actuate an oscillator via an output port to send perfectly timed Morse code. In addition, the ham radio operator might use the computer with an appropriate program to serve as a "contest logging aid." The "logging aid" would serve as an instant reference file whereby the operator could enter the calls of stations as they were worked and have the computer verify if the contact was a duplicate. The computer could do other tasks too, such as record the time of the contact by checking an external digital clock (or by utilizing a program that would enable the computer to be used as a clock within itself)!

And, the computer can be used to play numerous games with, such as tic-tac-toe, checkers, word games, card games, and a large variety of other types of games that one can program a computer to perform.

And perhaps most important — for the student, hobbyist, scientist, businessman, or anyone interested in the exciting possibilities of its applications — the contemporary microcomputer offers unlimited possibilities for the expression of individual creativity. For the development of computer programs can be an extremely creative, exciting and personally rewarding pastime and offers essentially limitless ways to exercise one's creative capabilities in developing "algorithms" that will enable the machine to perform desired tasks!

The electronic hobbyist can be kept occupied for years with a digital computer.

WHAT SINGLE ELECTRONIC MACHINE CAN BE USED TO PERFORM/CONTROL ALL THE FOLLOWING TYPES OF SERVICES?



- Send morse code
- Control repeater stations
- Operate as a calculator
- Receive/send/buffer data between a wide variety of communication devices
- Monitor instruments
- Control machines
- Sort/compile data
- Test other devices
- Play games

the SCELBI-8B MINI-COMPUTER CAN!

SCELBI COMPUTER CONSULTING, INC. - The company that pioneered in producing the small computer for the individual user with the popular SCELBI–8H, now brings you the new SCELBI–8B with increased capability!

Like the former SCELBI–8H, the SCELBI–8B is built around the amazing '8 0 0 8' "CPU-on-a-Chip" which has been revolutionizing the electronics world.

However, the NEW SCELBI–8B offers extended memory capability at reduced cost! It is directly expandable to 16,384 words of RAM/ROM/PROM memory. This increased memory capability now means the user has the potential in a small and compact computer to support compiler type languages, manipulate sizable data bases for business and scientific applications, and support a wide variety of programs including those that take advantage of external mass memory storage devices.

The NEW SCELBI–8B still retains the outstanding features of its predecessor. Decoding logic for 8 Output and 6 Input Ports is built into the basic computer. Plug-in capability for I/O devices is provided on the chassis. A unique, simple to operate console that utilizes just 11 switches on the front panel makes the SCELBI–8B a pleasure to use.

The NEW SCELBI–8B is backed by a line of low cost SCELBI interfaces which currently include: an interface that turns an oscilloscope into an alphanumeric display system, low cost keyboard and TTY interfaces, and an interface that turns a low cost audio tape cassette into a "Mag–Tape" storage and retrieval unit.

Last, but certainly not least, SCELBI has a wide selection of software ready to run on the NEW SCELBI–8B including: Editors, Assemblers, calculating programs, I/O and general utility routines. Additionally, SCELBI produces publications that can show you how to develop your own custom tailored programs.

The NEW SCELBI–8B is **available** NOW. (We have been delivering since June!) It is available in three forms. Ultra-low cost "Unpopulated" card sets with chassis kits starting at $259.00*. Complete parts kits for a 1,024 word mini-computer as low as $499.00*. An assembled and tested 4,096 word computer is just $849.00*. Interfaces, accessories, and software sold separately.

(*Domestic prices.)

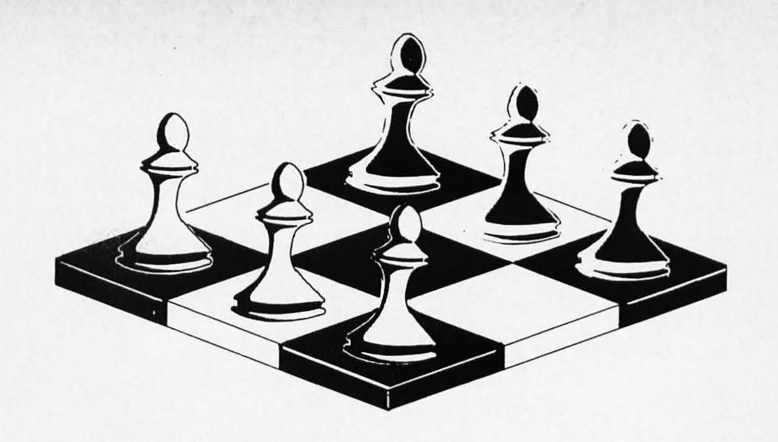
(Prices, specifications and availability subject to change without notice)

Literature available for S.A.S.E.

SCELBI COMPUTER CONSULTING INC.

1322 REAR BOSTON POST ROAD
MILFORD, CONNECTICUT 06460

HEXPAWN



A Beginning Project

in ARTIFICIAL Intelligence

What is intelligence? Pushing aside the philosophical and psychological questions for the moment, I can offer an operational definition of intelligence in programs: An "intelligent" program is one which was designed with a range of possible circumstances in mind, rules defining successful and unsuccessful responses to such circumstances, memory of the history of past responses and relevant circumstances, and an algorithm for using such past history information when similar circumstances occur again. Robert Wier has provided an example of a simple game application which illustrates this definition of intelligence in programs. Does it sound too deterministic for you? Hardly – the response is in some sense inherent in the program and its context. But, just as in natural life, the order and degree of the various inputs to the AI program cannot be predicted in advance with any great certainty. Just as each individual person is unique, each individual run of a good AI program will tend to differ – AI programs, like people, are good for lots of surprises.

by
Robert R. Wier
1208 Mistletoe Drive
Fort Worth TX 76110

Artificial intelligence. The very words themselves are at once frightening and fascinating. Hal lip reading; Colossus communicating with Guardian in a *real* "machine language"; M5 taking over the Enterprise. Yet these are still media creations, and we are cushioned by the comforting buffer of a movie or TV screen. To realize what artificial intelligence (or AI) is *really* like, you have to create it yourself (ever have an urge to play Frankenstein?). HEXPAWN originally appeared in *Scientific American* (Vol. 206, No. 3, p. 138) in Martin Gardner's "Mathematical Games" column. It is simplicity itself. The game board is identical to that of the standard two-dimensional tic-tac-toe, and two players control three pieces (or Xs or 0s or whatever) each. Each player's objective is to advance his pieces to the opposite side of the board, or eliminate or block the opposition's pieces. Moves of each piece are the same as the pawn in chess (i.e., move 1 forward to a vacant square, take diagonally).

HEXPAWN rules are very simple: To win, attempt to move one of your pieces to the opponent's side of the board, or block him from making any move. Moves are those of the pawn in chess. That is, you may move one square forward to an unoccupied square, or you may move one square diagonally in order to "take" an opponent's piece. Only these two moves are allowed. You may not move diagonally *without* a "take"; you may not move forward *with* a "take". Fig. 1 illustrates a typical game situation of occupied and unoccupied squares. In this "model" of the layout, the computer(X) can move in two ways which "take" the human pawn in the central square (number 4). The computer can move in one way which will not "take" a human pawn.

For a complete explanation, please refer to the original article (every library should carry *Scientific American*, and if yours doesn't, ask them why).

This version of HEXPAWN is a game that *learns.* You

may play it several times beating the computer (which keeps track of the board, as well as acting as one of the players) with ridiculously simple strategies. However, as play progresses, the computer notes its mistakes, and eventually, after 8 to 10 games, you may only tie or lose to the computer. The machine has "learned" how to play the game successfully.

The method described here to implement HEXPAWN is strictly brute force, and many techniques may be used to improve both the execution time and storage efficiencies. But in order to fully appreciate the internal workings of HEXPAWN, it is nice to keep it simple. Also, since this is a self-modifying program (a necessity in almost all AI), programmers will recognize that "simple is good," since after the code runs wild a few times and produces strange and wonderful results, it is fortunate to have code which is easy to debug.

HEXPAWN was implemented by the author on a 16-bit/word mini with an assembler. In this version it occupies 88E hex bytes, or 2190 decimal bytes, or 4218 octal bytes. It would be possible to reduce the memory requirement by using two or three bits instead of two bytes for the board representation of the playing pieces, but this would require a lot of bit diddling that is tedious unless you are really tight on memory. The minimum representation of the three states requires a two-bit binary number, using three of the four possible states of two bits. This requires only one word of memory. A less compact but easier to program bit level representation is to use three bits, one for each state. Only one bit would be "on" at any given time if the corresponding state is present. But on many computers it's considerably simpler to use two bytes so that pieces may be represented by "X", "0", and " " (space). The storage requirement will also vary considerably with the nature of the peripherals used, due to whatever interface programming is necessary. The original was implemented with a CRT where the cursor was "locked" in synchronization with a programmed counter notifying the program of the board location of the square being referenced.

Basically, the structure of the implementation is quite simple. In the *Scientific American* article, all possible board configurations are presented. Note that some are mirror images of others, but these are still required. These board configurations are hereafter referred to as "models". The program attempts to match the current board configuration with the models stored in memory. When a model is found, several courses of action may be available. In some cases, only one move will be possible, thus the computer is limited to that move. In other cases several moves are possible. The computer will select one (whichever is first on the list) and make the move. If a model is not found, this is an error situation; an illegal move has been made on *your* part, and an error message should be output. Fig. 2 is a "macro" flowchart of this process.

Following each model in memory is a string of move index bytes followed by a hex "FF". The "FF" is used as a terminator for that particular model. The bytes between the model and the "F" are index numbers for possible moves — the index references a jump table to produce a correct move by executing a jump.

A jump table is a very handy device when you need to reference several different

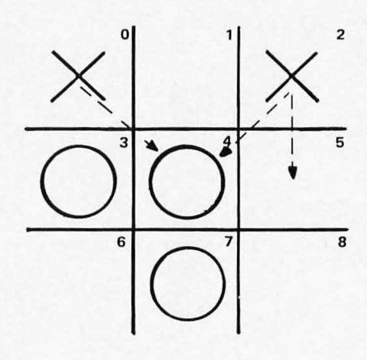
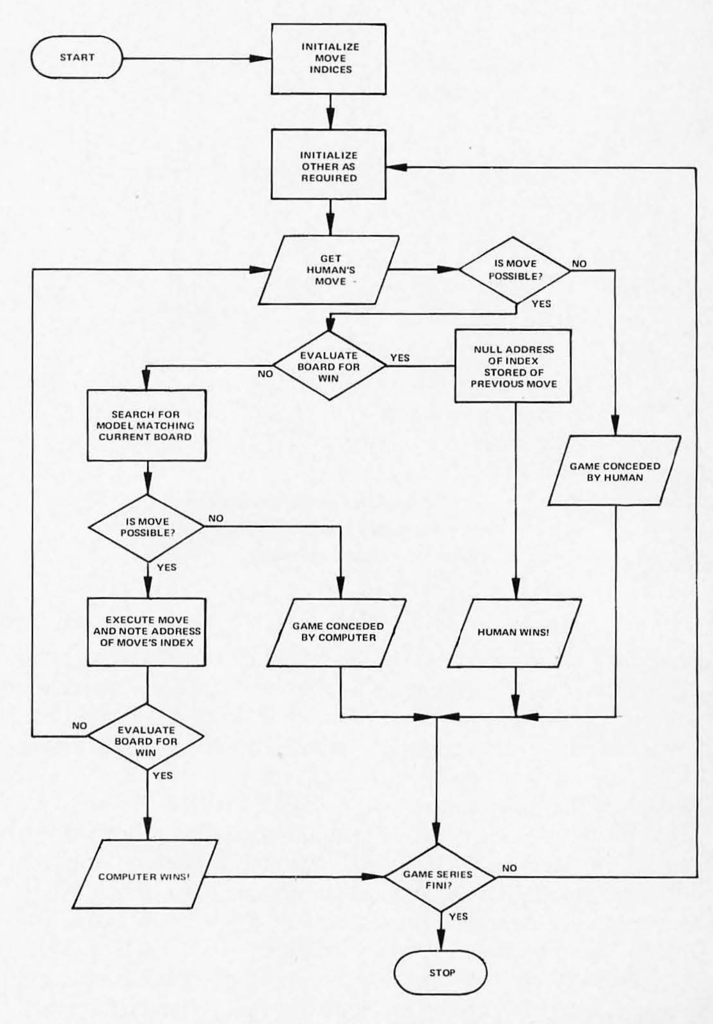


Fig. 1. The game layout for a typical move.



Note that the move indices are initialized only once for each series of games. Initialization for each game will defeat the learning process.

Fig. 2. Control flow logic for the HEXPAWN program.

Fig. 3. Table of All Possible Moves (Models).

	BOARD POSITION MODEL SQUARE 0	1	2	3	4	5	6	7	8	COMPUTER'S POSSIBLE MOVES (see Fig. 4)
	X	X	X	0				0	0	3, 4, 7
	X	X	X			0	0	0		1, 4, 5
	X	X	X		0		0		0	1, 2
	X		X	X	0				0	2, 6, 8
		X	X	0	X				0	3, 7, 11
EXAMPLE OF FIG. 2	X		X	0	0			0		2, 6, 7
	X	X		0		0			0	3, 4, 5
		X	X		X	0	0			5, 10, 11
		X	X	X	0	0	0			5, 6
	X		X	X		0		0		8, 9
	X	X		0	0	X			0	2, 3
		X	X	0		0	0			3, 4, 5
		X	X		0			0		6, 7
		X	X		0		0			6, 7
	X		X	0				0		7
			X	X	X	0				8, 11
	X			0	0	0				2
		X		X	0	0				8, 5
		X		0	0	X				3, 14
	X			X	X	0				8, 11
	X		X	0		0			0	15
	X			0	X			0		15
			X	0	X	X				11, 14
			X	X	0					6, 7, 8
		X		0	X					3, 11
		X		X	0					5, 11
	X		X	0						2, 8
			X		0	X				6, 14
	X			0	0					2
	X		X		0	0		0		1, 2, 6
		X			0					15
	X		X	0	X	0		0		15
			X	0	0	0				6

Key: **X = computer piece occupies square**
0 = human's piece occupies square
blank = square is empty

locations in your program using numerically sequential indices. The advantage is that after assembly, debugging is facilitated. If you desire to change all the jump addresses of a particular segment of code, you need only change the jump table, rather than each reference containing the desired jump address. It is also unnecessary to worry about having to make the code referenced in the jump table equal in length. All that is taken care of in the jump table itself in an easy and consistent manner. The jump table is particularly appealing in that you have multiple-level-indirect addressing capability.

HEXPAWN learns by removing the index which leads to a defeat for the computer. Thus, if a move to square 8 results in a loss, the index following the appropriate model is changed to a null character, which *eliminates the losing move.* It is easily seen that if a particular move always leads to a loss, it will be completely nulled, thus allowing the computer to "know" several moves ahead that it has lost the game. As each losing move's index is nulled, the learning process effectively progresses toward earlier moves.

As noted in the original article, this version only penalizes the *losing* move. Also of possible consideration is the rewarding of a winning move, but this would complicate our code considerably.

For convenience's sake let us number the squares of the playing board 0-8 starting in the upper left corner, working horizontally and down. Let us also establish the convention that the human player always moves first. This does not seem to compel a deterministic game. That is, either player may win regardless of who moves first. Now suppose that there has occurred a particular board configuration (Fig. 2). Note that the "X" pieces belong to the computer, while the "0" pieces are yours. You have just made the preceding move, and now the computer must decide what to do. The computer's possible moves are indicated by the dotted lines. But how does the computer know this? It searches through memory until the following bit pattern is found (in hex):

E740E7D6D64040D640020607FF

The first 9 bytes represent the board. Note that in EBCDIC, E7 is an "X", D6 is an "0", and 40 is an " ". Remember that these are EBCDIC codes (my peripherals use it), but it could just as easily have been ASCII. The next three bytes represent the indices for possible moves as they exist at the beginning of the game. That is, the possible moves are these:

02: "X" in square 0 moves to square 4 taking your "0"

"To realize what artificial intelligence is *really* like, you have to create it yourself . . . "

06: "X" in square 2 moves to square 4 taking your "0"

07: "X" in square 2 moves to square 5

Now, either move 02 or move 07 will result in a loss the next move that "0" makes (assuming that you are trying to win) and *that* index will be nulled so it cannot be selected again in the event of this same board configuration. Move 06 is correct since it removes your piece and also blocks you from obtaining "X's" side of the board. Since the computer simply selects the first move on its list, the first time this board configuration is encountered, the computer will lose (as a result of move 02). However, after this game the computer will select move 06, which is correct, since it is next on the list. The number of the index has no particular significance; it could be anything as long as it denotes the displacement needed in the jump table to direct the flow of control to the proper code for the move desired. The "F" is a terminator that signals the end of that particular model and move list.

We will not present the actual code to accomplish the HEXPAWN algorithm since there are so many machines of a differing nature in hobby use. However, copies of the author's LOCKHEED SUE Minicomputer version are available from him for $3 to cover the most of duplication and postage.

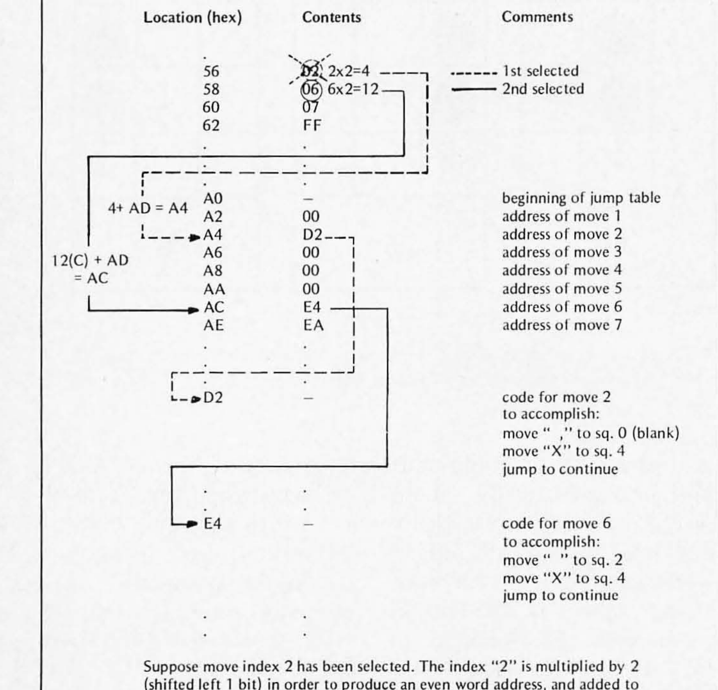
A few hints are in order to help you avoid some of the more obvious problems. The

EXAMPLE 1: To illustrate, assume that the following is in memory at the start:

	Location (hex)	Contents (hex)		Comments
	.	.		
Step 1.	56	02 ← Step 6.		
	58	06	If loss store	move indices after appropriate
	5A	07	"00" here.	model
	5C	FF		
	.	.		
	.	.		
	A0	XX		beginning of jump table
Step 2.	A2	XX		
	A4	D2		address of move 02
	A6	XX		
	A8	XX		
	AA	XX		
	AC	E4	Step 4.	address of move 06
	AE	EA		address of move 07
	.	.		
	.	.		
	D2	XX		code for move 02

The "learning" sequence is composed of the following steps:

1. Search models until match is found.
2. Select first index of possible moves, add to location of beginning of jump table, giving location of address of that move's code – in this case, index 02 x 2 (to get even byte boundary) + A0 = A4. If no possible move (no non-null index) is available, concede game to human player.
3. Note the address of the index used – in this case "56."
4. Jump using indirect addressing to the move's code and execute – in this case, location D2.
5. Evaluate board for win or loss.
6. If loss has occurred, null the location of the last index used – in this case "56", thereby removing this move from the machine's repertoire of responses to this *particular* board configuration. If a tie or computer win has occurred, do nothing to the index.



biggest hang-up with this program is to get it running correctly in regard to the jump table. If a wrong index is obtained, the program will run off into the boondocks and never be heard from again. Therefore it is nice to include checks on the validity of the index retrieved and to output an error message in the event of something strange happening. A reasonable board may be printed using dashes and exclamation marks. However, if you do this, you will have to "unpack" the board as represented in memory into a more suitable form for I/O. If you don't have a CRT with machine programmable cursor, you can use the numbers assigned to the squares to indicate your moves. Of course you'll want the machine to have a variety of responses for being blocked, losing and winning. For debugging it is good to output the number of the index which is nulled after a losing game. In this way you may keep track of the learning process as it advances. Also you should be aware that if the human player makes some illegal moves, no model will be found, and a message should be output indicating this fact.

But not *all* illegal moves will result in an error condition. In this case, should the human player win, the machine will null the last move's index *even if it is correct.* After this happens a few times, the machine will start making illegal moves, acting illogically, and generally approximating a nervous breakdown!

Programming HEXPAWN will painlessly (?) introduce you to a number of worthwhile aspects of the logical arts. You'll see that many segments of code (such as the board evaluation) are similar and are potential

---

In a written communication, Martin Gardner points out that the original Hexapawn article is reprinted as Chapter 8, "A Matchbox Game-Learning Machine" in his book The Unexpected Hanging and Other Mathematical Diversions *(Simon & Schuster, 1969). The book version includes updates of the drawings in the original* Scientific American *article, notes on reader reactions to Hexapawn, and reference to an article on the more general game "Extendapawn." Our thanks to Martin Gardner for his assistance in supplying a corrected version of Fig. 5 for use in BYTE.*

Fig. 4. Table of Computer's Moves ("X" Graphic).

COMPUTER'S (X's) MOVES

MOVE INDEX #	SQUARE TO SQUARE		COMMENTS
1	0	3	
2	0	4	
3	1	3	
4	1	4	
5	1	5	
6	2	4	
7	2	5	
8	3	6	computer wins!
9	3	7	"
10	4	6	"
11	4	7	"
12	4	8	"
13	5	7	"
14	5	8	"
15	–	–	computer blocked

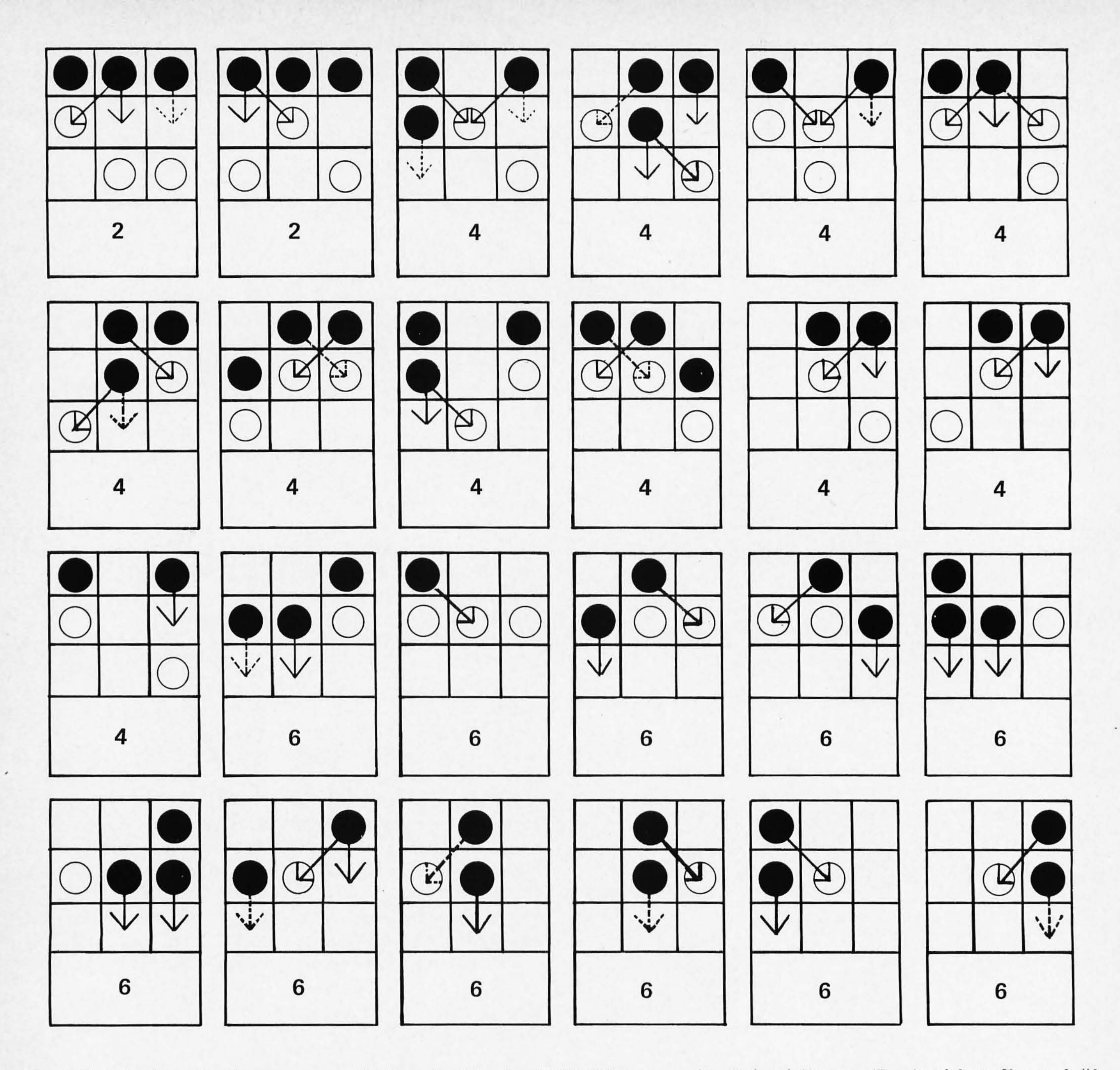


Fig. 5. The set of possible Hexapawn game situations faced by the HEXPAWN program after 2, 4 and 6 moves. (Reprinted from Chapter 8, "A Matchbox Game-Learning Machine," in The Unexpected Hanging and Other Mathematical Diversions *by Martin Gardner.)*

A BASIC Version of This Program:

For those with systems running the BASIC language, a BASIC version of this program called HEX is found on page 122 of the third printing of *101 Basic Computer Games,* available for $7.50 + 50¢ postage from Digital Equipment Corp., Software Distribution Center, Maynard MA 01754.

candidates for subroutines. You'll see that indirect addressing does indeed have some practical uses, if you can ever get the code debugged. You'll see that it is *very* important to try and anticipate possible sources of error in your code before you run the program, and at least to include a mechanism to warn you when problems occur. (I didn't anticipate any problems with the jump table and consequently spent several hours trying to figure out how the move indices were coming up with such strange values. If I had put in some code to check them first, this process would have been shortened considerably.) You'll see that some programs are complex to such a point that you simply cannot sit down and write them without thinking about the logical design first! You'll see why you should *never, ever* write programs that are self-modifying in nature (except AI, naturally).

Lastly, amaze (antagonize) your friends by sitting down at your computer and winning four or five games, then inviting them to try. When they can't, you can smile smugly and explain how your computer learns from its mistakes, and so should they!

COMPUTER EXPERIMENTER SUPPLIES

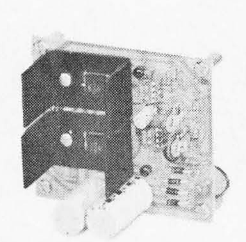
FACTORY FRESH—PRIME QUALITY
PERFORMANCE GUARANTEED

MICROPROCESSORS AND MEMORY

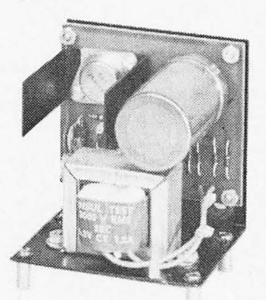
8008	$ 35.00	Commercial Grade—up to 35° C.
8080	135.00	These units are factory fresh, full spec devices.
2102	3.50	
2102-2	4.50	

COMPUTER GRADE REGULATED POWER SUPPLIES

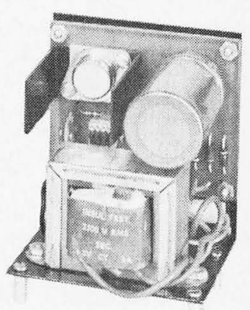
All units are short-circuit proof, fold back current limited and with over-voltage crowbar protection.



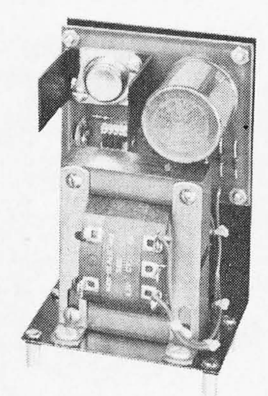
MD-15
±15 Volt at 200MA
Dual Tracking
$30.00



MD-5-1
+5 Volt at 1 Amp
$24.50



MD-5-3
+5 Volt at 3 Amp
$34.50



MD-5-6
+5 Volt at 6 Amp
$44.50

MICRO COMPUTER SUPPLY COMBINATIONS

For the 8008
MD-08—+5 volt at 6 amp, -12, -9 at 200 ma $75.00
For the 8080
MD-80—+5 volt at 6 amp, ±12v at 200 ma ... $75.00
For the Fairchild F-8
MD-8—+5 volt at 6 amp, +12 v at 200 ma ... $65.00
For the M6800
MD-5—+5 volt at 6 amp $44.50

All units are short circuit proof, fold-back current limited and with over voltage crowbar protection.

TTL INTEGRATED CIRCUITS

All devices are factory fresh, full spec units.

7400	.23
7404	.25
7442	.60
7447	.95
7448	.95
7475	.60
7490	.60
7493	.60
74125	.55
74126	.55
74192	1.10
74193	1.10

GUARANTEE

Most devices shipped within 24 hours. If not shippable within 2 weeks payment refunded. Performance guaranteed on all units for 30 days. Defective parts replaced at no charge. NOTICE: This warranty applies **only** to parts that have not been soldered. You must use sockets for your incoming inspection tests.

All Prices Subject to Change Without Notice
Minimum Order $10.00
Add $1.00 to Cover Postage and Handling
Send Check or Money Order (No C.O.D.) To:
N. J. Residents Add 5% Sales Tax

MICRO DIGITAL CORP.

BOX 413, EDISON, NJ 08817 • (201) 549-2699

Computers And Amateur Radio

Time-sharing by radio? Radio packet switching networks? Program exchange meeting grounds in the high frequency bands? Computer controlled ham radio stations? Read on . . . Mike Gipe provides us with this article on the synthesis of amateur radio and computer hobby activities into a combination which is more fun than the simple sum of parts. While there are no bureaucratic restrictions on home microcomputer systems, there are some federal regulations you must comply with in order to become a ham. For many computer hobbyists, the two-hobby combination would be well worth considering – despite the required ham license exam. Mike provides several references to more detailed information for those individuals who want to check out amateur radio.

by
Michael A. Gipe WA3GAU/1
155 Bay State Road
Boston MA 02215

Computer construction and programming is the newest hobby in the field of electronics; ham radio is undoubtedly the oldest. The fascination of electronics is certainly a good reason why many people are interested in both computers and ham radio, but it is not the only reason. The two hobbies are complementary. The person who spends his leisure time on both will surely find that one benefits the other, making it more fun. Computers handle information – they receive it, deliver it, condense it, modify it, utilize it and sometimes even mangle it. The versatility of the computer is reflected in the wide variety of information that it is called upon to digest. Ham radio is a hobby dedicated to the art of communication – the transfer of information. Without information, there is no need to communicate, and without communications, the generation of information is useless. (What good is a computer without any I/O?) Obviously, computer buffs find ways to communicate without ham radio, and hams have never been speechless because they didn't have computers. However, marrying the two offers the opportunity to communicate information (of all kinds) on a much wider scale. The computer buff who includes radio in his field of interest will certainly find his hobby more challenging and more rewarding.

It is no surprise, therefore, that many of BYTE's readers are also amateur radio operators. But it is likely that a number of BYTE's readers are not familiar with amateur radio. This article is an introduction to the hobby for these people. Hopefully, it will also suggest some new ideas for those who already have discovered that ham radio and the computer make an excellent pair.

What is Ham Radio?

Radio amateurs are authorized to transmit and receive signals with their own radio stations. This makes it possible to experiment with many different means of radio communication and to converse with all kinds of people from all parts of the world. The fact that ordinary people can make important discoveries while pursuing a hobby is demonstrated by a number of amateur accomplishments including: transatlantic communication at high frequencies, moonbounce communication, the practical use of single sideband transmission (SSB), and the increase of our understanding of radio propagation through the atmosphere. The fact that amateur radio is for personal use is stipulated by international agreement and federal regulations which enjoin any amateur from exploiting amateur radio for monetary purposes. Amateur radio is a fine hobby; it's fun and educational.

What's Happening

Amateurs have been allotted many segments of the radio spectrum beginning as low as 1.8 MHz. Table I shows the frequency bands available. All types of modulation can be utilized; however, certain types are restricted to selected frequencies for bandwidth and interference reasons. Computer hobbyists will be



The typical (?) ham radio station has various pieces of equipment. For low band work in "single sideband" (SSB) voice communications, the minimum equipment is a transmitter, a receiver (or combination), plus some form of antenna outside on your roof. Such a setup might cost $500 in today's markets – although less expensive radio shacks might be set up by buying used equipment with the guidance of experienced hams in your local amateur radio club.

The real fun starts when you add a computer.

---

interested to learn that radio teletype is an often used mode. Any person may be authorized to use any frequency and mode available to amateurs or only certain portions, depending on the class of license which he holds. This will be explained later in detail.

Hams have put these resources to a variety of uses, some of them quite exciting. By connecting his receiver and transmitter to the telephone line through a "phone patch," a ham can make long distance phone calls which might not be otherwise possible for economic or technical reasons. Thousands of soldiers in Viet Nam were able to talk to their families this way. Amateurs also serve the public by providing communications in emergency situations such as fires and floods. It is possible to put an amateur radio station in a car and operate "mobile". Two recent developments have stirred up a great deal of interest: repeaters and satellites. Amateurs have designed and built seven operational earth orbiting satellites since 1961. Of course, NASA helped put them in orbit. The sixth and seventh of the series of satellites, known as OSCAR for "orbiting satellite carrying amateur radio," are now

Band Designation (wavelength)	Frequency Range (MHz)
160 meters	1.8 – 2.0
80 meters	3.5 – 4.0
40 meters	7.0 – 7.3
20 meters	14.0 – 14.35
15 meters	21.0 – 21.45
10 meters	28.0 – 29.7
6 meters	50 – 54
2 meters	144 – 148
	220 – 225
	420 – 450
	1215 – 1300
	2300 – 2450
	3300 – 3500
	5650 – 5925
	10000 – 10500
	21000 – 22000
	40000 and up

Table I. Communications frequencies available to amateur radio operators.

Ordinary people have made important discoveries in the radio field as amateurs – and you can expect similar things to happen in the computer field as more and more people experiment with the technology.

relaying amateur signals great distances around the globe. Another device used to overcome the line-of-sight distance limitation for VHF and UHF frequencies is the repeater. These receive signals and retransmit them at a different frequency. By installing repeaters at high places like mountaintops and tall buildings, hams can communicate over a wide area with small transmitters. In New England, for example, repeaters have made it possible to talk with people in several states using only a battery-powered walkie-talkie. By installing radio equipment capable of accessing a repeater in his car, a ham can have companionship or emergency aid available at the touch of a button whenever he is driving. A number of frills are also possible with a repeater. Some repeaters have been set up with a telephone line tie-in. This telephone line can be used to make telephone calls from nearly anywhere. Imagine the possibilities – while driving to your girlfriend's apartment you can hit a few buttons on your mobile radio and call your wife on the telephone to tell her that you have to work late!

More computer-oriented hams are needed to push for changes in the regulations to facilitate the computer/radio synthesis.

On To The Good Part

The real fun starts when you add a computer. Your microcomputer (or your mini if you're rich) can be a very useful addition to the ham radio station. It's a great help in calculating orbit data for OSCAR satellite work. The computer could be programmed to figure out when and where the satellite will appear in the sky, ring a bell to warn you, turn on the transmitter and receiver, and point the antenna in the proper direction! A microcomputer can do a very nice job translating and generating Morse code. Along a similar line, teletype code and speed conversions are easily accomplished by a microcomputer. A few months of thought should give you dozens of ideas for using the computer in the radio station.

The combination of repeater and computer can be powerful. One use for the computer is to control the repeater. Regulations require certain controls on repeater operations. These regulations have undergone substantial changes recently and may change again, so the best way to get the details is to contact one of the sources mentioned at the end of this article. The computer can make it possible to add many very advanced, very fancy features to the repeater. Services beyond the telephone tie-in previously mentioned are possible. Another way to use the computer with a repeater is as a shared data processor. A mini or microcomputer could be accessed by radio through the repeater by many people. Any ham with a teletype could have access to a computer. Since the possible benefits are so great, it is inevitable that a repeater-computer combination will soon be in operation.

These applications of the computer to ham radio are interesting, but the long distance data communications made possible with ham radio may turn out to be the most valuable result of combining the two hobbies. For the individual computer owner, radio communications means being able to operate the computer remotely from a car, in the next town, nearly anywhere. Moreover, communication between computers is also possible. A large network similar to the ARPA network could be formed, opening the way for sharing programs and ideas. Hobbyists' efforts in this area will supplement current research in radio networking. With computers, nearly anything is possible.

Rules & Regulations

Radio is, by its very nature, an international concern. Periodically, representatives from most of the countries of the world meet and draw up voluntary agreements concerning the use of one of our valuable resources, the radio spectrum. Amateur radio has its foundation in these agreements. It is defined, governed and allocated frequency space. In the United States, the Federal Communications Commission (FCC) writes the rules and regulations, consistent with international agreement, which govern ham operation here. It may be of interest to point out that hams are regulated by law, but computer hobbyists are not. The reasons for regulating radio should be fairly obvious.

The rules for ham radio are much more liberal than those for other services. Amateur radio is intended to be flexible and to allow experimentation. According to the international agreement, *though, hams* must be licensed. The FCC issues licenses and also enacts and enforces the regulations which licensees must follow. The FCC determines what modes of transmission can be used at what frequencies and who gets to use them. Full details about the regulations can be found in the publications listed at the end of this article.

Obviously, computer buffs find ways to communicate without ham radio, and hams have never been speechless because they didn't have computers.

Limitations

Although the rules are designed to be flexible and allow experimentation, sometimes they do not keep pace with the activities currently underway. Some regulations may inadvertantly restrict some harmless operations. Computer applications in ham radio are new and there are a few regulations which might concern the computer hobbyist-radio amateur. Although teletype transmission is allowed on nearly every band, the speed and type of transmission is restricted. Currently, only the five bit Baudot code may be used and the maximum rate of transmission is 100 words

per minute (75 baud). ASCII code is not permitted. Since most amateur teletype operation previously was with surplus Western Union equipment, this was no problem. Now, however, it conflicts with computer industry standards. Until the rules can be changed, the microcomputer can be used to convert from one code to the other. One other restriction which may be useful information is that full duplex operation is not permitted. Other aspects of the regulations may dictate certain techniques, but in general the computer hobbyist should find that it is possible to do what he wants, somehow.

The rules can be changed to make things easier, however. Any petition to the FCC will be considered. More computer-oriented hams are needed to push for these changes.

How To Get a License

What are the details about licenses? There are actually two licenses issued by the FCC. One, the operator's license, permits the holder to operate [illegible] radio station. The other is a permit to set up a station at a permanent location. If you wish to set up more than one *permanent* station, a station license must be obtained for each. However, one station license allows you to set up one permanent station and any number of mobile or portable stations. A set of call letters issued with the license serves to identify the station on the air. The two licenses are distinct but are actually printed on one piece of paper.

According to the international agreement, a person must demonstrate the ability to send and receive Morse code to get a license. He must also demonstrate some knowledge of elementary radio theory and operating techniques. The FCC conducts examinations for licenses. These are not difficult to pass with a little study, and excellent aids for learning code and theory are available. See the references for some of these.

The license structure is further complicated by different classes. As an incentive to increase skill and knowledge, more operating privileges are granted to those who demonstrate greater proficiency. Each class of license represents a specific set of privileges and a correspondingly difficult examination. There are currently six classes but changes may occur within the next year. The Novice class is the simplest. The exam is very simple and the privileges are few, but it is an excellent way to get started. The Novice class license requires a 5 word per minute Morse code proficiency. Next is the Technician class. This class is designed for those who wish to do experimental work on VHF or UHF frequencies. Only frequencies above 50 MHz may be used by the Technician. The code test is the same as for the Novice; the theory test is the same as the General class. The Technician class license may be the best way for a computer hobbyist who wants to work with repeaters to get started. The General class license grants its holder permission to operate in a large portion of all bands and all modes. The Conditional class is the same as the General but allows a different application procedure for special hardships. The Advanced class grants more frequency space reflecting the additional technical proficiency needed to pass the exam. The highest class is the Extra class which awards all amateur privileges. It's a complicated system, but it makes it easy to get started and it rewards those who improve their skills.

Actually getting your license is as simple as any bureaucratic operation. You must fill out an application, pay a small license fee, and take the tests. Most tests are conducted at local FCC field engineering offices. More details can be obtained by calling the nearest office. The number is in the phone book.

After passing the exams and waiting the usual bureaucratic waiting period, you should receive your license. It is good for five years, and may be renewed simply by mailing in an application and a check. The fun is well worth the effort.

Hopefully, this has stimulated a little thought about information and communication. When computers and ham radio are teamed up, the possibilities are limited only by the imagination. Computer hobbyists who are looking for an additional challenge should look at ham radio.



License study guides are available for each class of amateur license grade. These were written by the 73 staff and are available from Radio Bookshop, Peterborough NH 03458. Morse code tapes are also available geared to each license grade.

50¢

73

FCC RULES AND REGULATIONS

Part 97 Amateur Radio Service

published by
73 MAGAZINE INC Peterborough NH 03458

Part 95 of the *FCC Rules and Regulations* governing amateur radio is available from:

73 Magazine (50¢)

Although ham radio rules are designed to be flexible and allow experimentation, sometimes they do not keep pace with the activities currently underway . . .

Excellent educational aids are available from:

American Radio Relay League, Inc.
Newington CT 06111

and

73 Magazine
Peterborough NH 03458

The Digital Feedback Loop

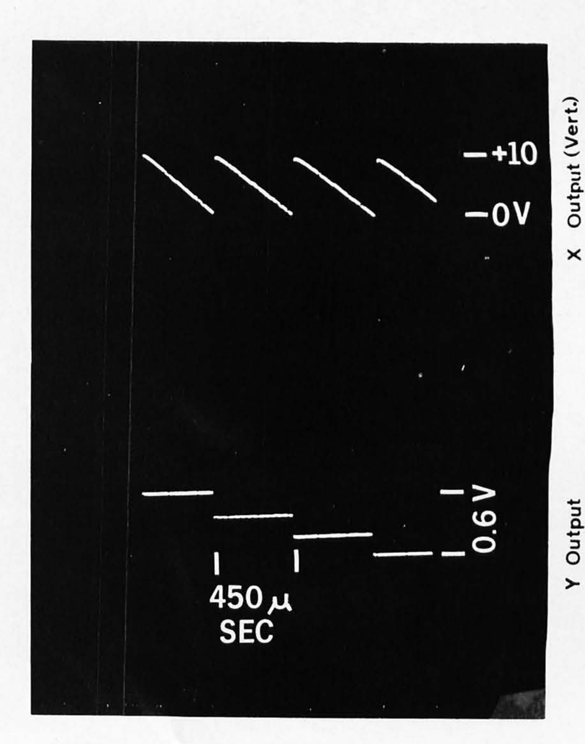
Sumner S. Loomis, proprietor of Loomis Laboratories, Route 1, Box 131A, Prairie Point MS 39353, provides this set of pictures represented in his version of the Hogenson oscilloscope graphics display design printed in BYTE #2. Mr. Loomis built the graphic interface using a preliminary version of the PC board for the scope display, but with one modification: he used CMOS instead of TTL integrated circuits. The following are his comments on the scope display as implemented in CMOS:

I found that adjustment of R1 and R2 pots can cause the DACs (IC15 and IC16) to overheat. I believe that this is why one of my DACs now exhibits a slight overlap in the digital ramp (bit 4 is off value). I would recommend that these pots be set at maximum resistance setting, and upon initial setup the pots should be advanced while observing the respective output ramp on an oscilloscope. Decrease the resistance setting until the maximum height ramp (without clipping), is obtained. The overheating problem occurs with severe clipping, which happens at the low resistance setting.

I found it necessary to change IC17 to an LM318, which has better high frequency response for the fast sweep rate of this ramp. My conversion of the digital ICs to CMOS was also successful, without a hitch. I am very pleased with the low power drain and logic swings from rail-to-rail that result from this change. The substitution was accomplished as follows:

IC1 becomes 74C10
IC2 becomes 74C04
IC3 becomes 74C00
IC4 becomes 74C00
IC5 uses 7408 TTL (no 74C series equivalent)
IC6 becomes 74C155
IC7-IC10 become 74C193

I am going to have to change the Z-axis op amp in my display unit to accommodate the high frequency drive. The present amplifier is too slow, and causes some smearing of the dots as well as low brightness on lone dots. Everything else checks out OK!



X & Y OUTPUTS (STAIR STEP RAMPS)

Fig. 1. Staircase ramp wave forms for X and Y drives. This picture illustrates the stepping of the X and Y drive DAC outputs during several sweeps of the display raster.

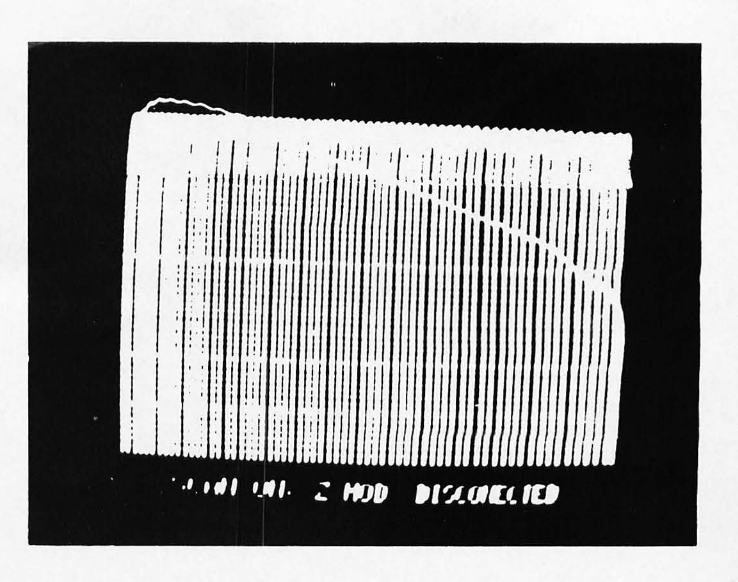


Fig. 2. A full raster achieved by turning on the scan and disconnecting the Z modulation input from the scope.

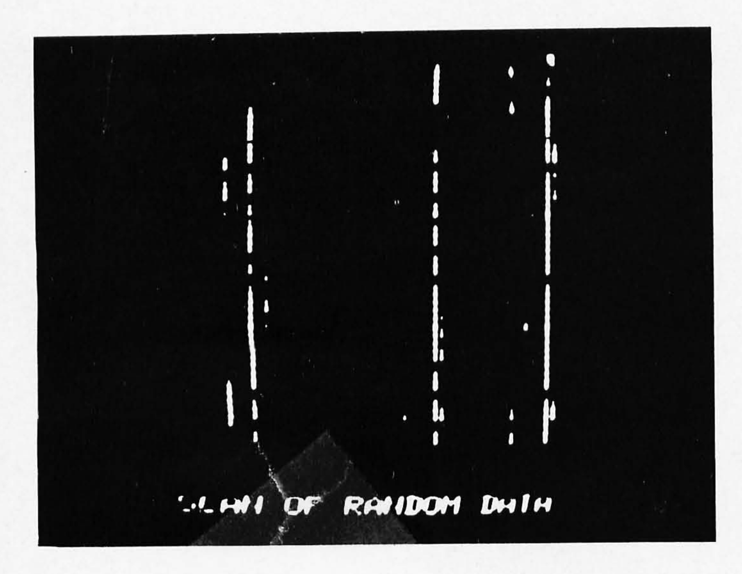


Fig. 3. A pattern of random data acquired when the scope interface is turned on.



Fig. 4. A pattern of horizontal dots programmed into the display by hand.

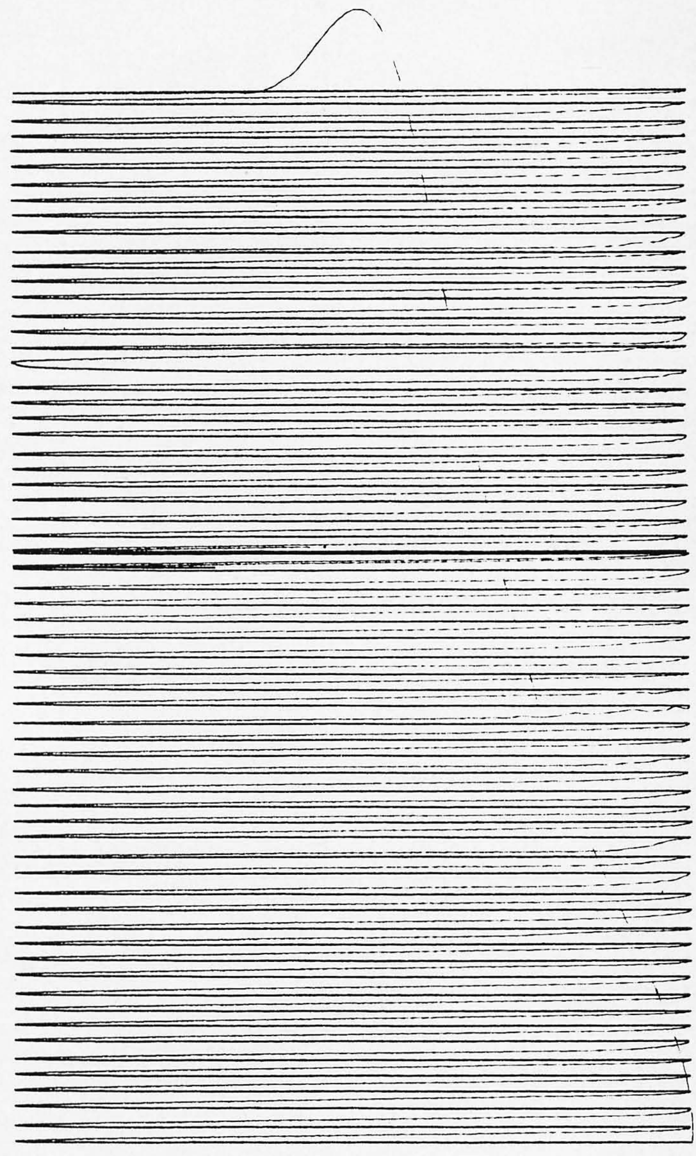
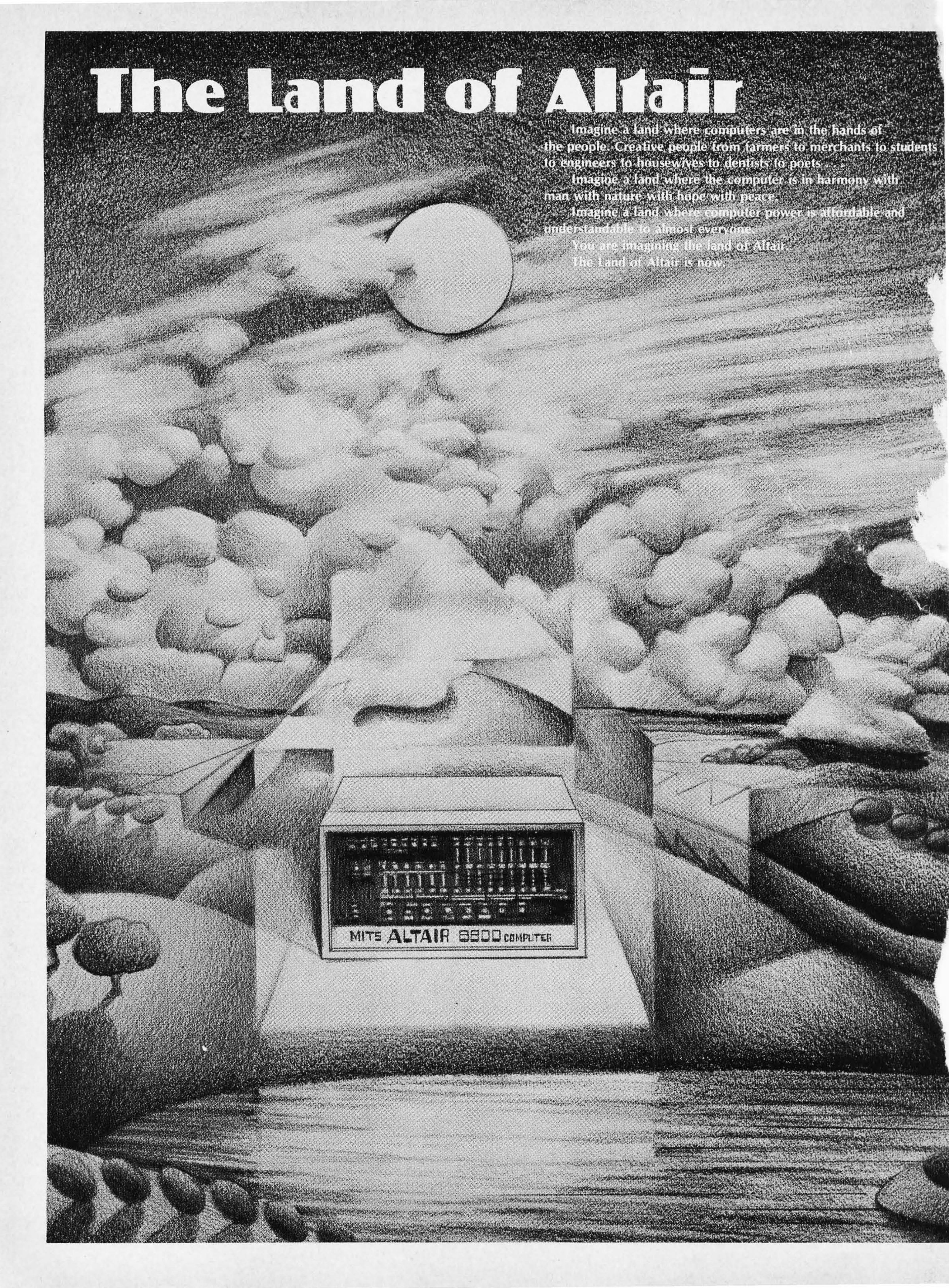
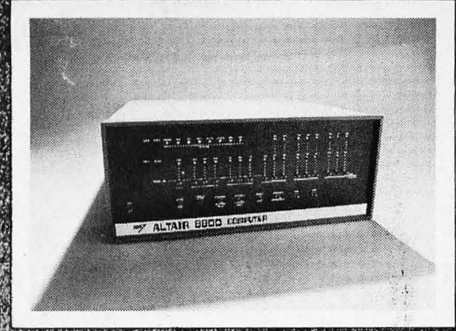


Fig. 5. Mr. Loomis used an interesting trick to achieve the printout in this illustration – he changed the main timing capacitor of the 555 oscillator to 5 microfarads and used the much slower sweep to drive an X-Y plotter. In order to get hard copy printouts (albeit slowly) about all one would have to add is a jury rig "intensity" modulation input to raise and lower the recorder pen under control of the interface. Refinements such as superimposed small amplitude X-Y oscillations could be used to draw small lissajous-patterns (circles or figure eights) at each "on" point of the hard copy.


The Land of Altair
Imagine a land where computers are in the hands of the people. Creative people from farmers to merchants to students to engineers to housewives to dentists to poets . . .
Imagine a land where the computer is in harmony with man with nature with hope with peace.
Imagine a land where computer power is affordable and understandable to almost everyone.
You are imagining the land of Altair.
The Land of Altair is now.
MITS ALTAIR 8800 COMPUTER

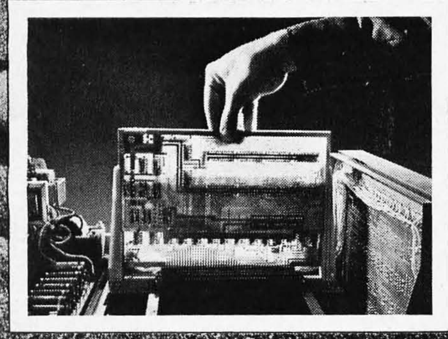


The Altair 8800 is a powerful, general purpose computer that sells for an amazingly low price of $439 in kit form and $621 assembled.

The Altair 8800 is a superbly engineered, variable word length computer. Its byte orientation structure was designed to give the Altair the most efficient utilization possible — an efficiency only found in the most advanced computers.

The Altair 8800 has bench marks comparable to those of much more expensive mini-computers. It has a cycle speed of 2 microseconds; it can directly address 65000 words of memory and 256 input/output devices; and it has over 200 basic machine instructions.

The Altair 8800 is backed by an extensive software development program. Altair BASIC and EXTENDED BASIC language are designed for most computer needs from business to home to scientific applications. Other software includes an Assembler, Text Editor, System Monitor, Debug, and Disk Operating System.



The Altair 8800 has been designed with buss orientation to be easily expanded and easily adapted to thousands of applications. Any card can be plugged into any slot and the correct address, etc., for that card will be picked up off the buss system.

The Altair can be custom assembled to fit almost anywhere. Besides general purpose computing (business, scientific and home uses), the Altair is ideal for process control and industrial uses.

The basic Altair includes Central Processing Board, front panel, power supply (enough to power any additional boards), and expander board (with room for 3 extra boards) all enclosed in a handsome, aluminum case. Up to 16 cards can be added inside the main case (memory and I/O boards).

Memory board options include 1024 word, 2048 word and 4096 word boards. Each memory board has memory protect features.

Interface board options include a Parallel Interface Board and 3 Serial Interface boards. These boards allow you to connect the Altair to our growing list of input/output devices.



For cost sensitive applications and for hobbyists, the Altair 8800 and most Altair options come in kit form. Already, thousands of Altair kits have been assembled and are in full operating order.

Altair kit builders include individuals, schools, companies, small laboratories, and industrial users.



The Altair Floppy Disk can store over 300,000 bytes of information on a flexible disk. With a data transfer rate of 250,000 words/second and a track to track access time of 10 msec., it has the capability for advanced data processing procedures.

Other Altair options include the Comter II Computer terminal with built-in audio-cassette interface for inexpensive mass storage, ASR-33 Teletypewriters, and the Altair 110 Line Printer which produces 80 columns of 5x7 dot matrix characters at 110 characters per second or 65 lines per minute.

PRICES:

Altair Computer kit with complete assembly instructions $439
Assembled and tested Altair $621
1,024 word memory card . . . . $97 kit and $139 assembled
2,048 word memory card . . . . $145 kit and $195 assembled
4,096 word memory card . . . . $264 kit and $338 assembled
Full Parallel Interface card . . . . $92 kit and $114 assembled
Serial Interface card (RS232) . . . . $119 kit and $138 assembled
Serial Interface card (TTL or teletype) . . . . $124 kit and $146 assembled
Expander Card (adds 4 slots to 8800) . . . . . . $16 kit and $31 assembled

SOFTWARE PRICES:

Altair 4K BASIC $350
Purchasers of Altair 8800, 4K of Altair memory, and Altair I/O card .. ONLY $60
Altair 8K BASIC $500
Purchasers of Altair 8800, 8K of Altair memory, and Altair I/O card .. ONLY $75

Contact Factory for complete Altair Price List.

Altair Documentation Special . . . . . includes Assembly manual, Operators manual, Theory of Operation manual, BASIC language manual, catalog and sample Altair Users Group newspaper . . . . $15 (offer expires November 1, 1975)

Ordering Instructions

You can order the Altair Computer by simply filling out the coupon in this ad or by calling us at 505/265-7553 or 262-1951. Or you can ask for free technical consultation or for one of our free Altair System Catalogs.

Prices, specifications and delivery subject to change. Warranty: 90 days on parts for kits and 90 days on parts and labor for assembled units.

MITS
Creative Electronics

MITS/6328 Linn NE/Albuquerque, New Mexico 87108, 505-265-7553

ALTAIR COUPON

☐ Enclosed is check for $ ________
☐ BankAmericard # ________
☐ or Master Charge # ________
☐ Credit Card Expiration Date ________

☐ Altair 8800 ☐ Kit ☐ Assembled
☐ Options (list on separate sheet) include $8 for postage and handling
☐ Altair Documentation Special
☐ Please send free Altair catalog

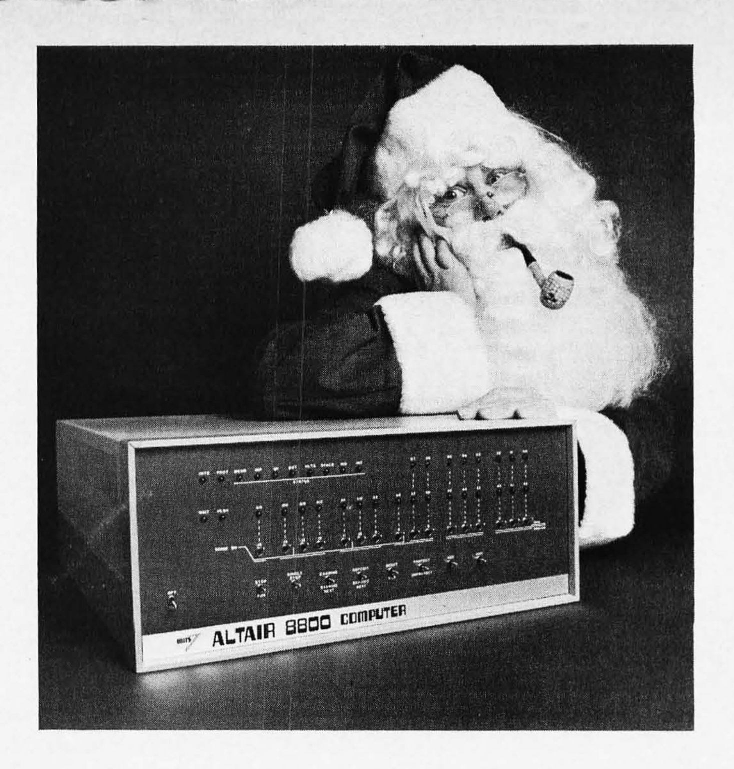
Name ________________

Address ________________

City ________________ State & Zip ________________

MITS/6328 Linn NE, Albuquerque, NM 87108, 505/265-7553 or 262-1952

If Santa had an Altair®...



Santa might be possible!

Have you ever wondered how Santa keeps track of all the addresses of all the children in the world? How he knows who gets what?

With an Altair 8800, Santa might have a fighting chance. He might be able to keep up with the ever-expanding toy industry. He might be able to remember who's been naughty and nice.

With an Altair 8800, Santa might be possible.

This Christmas you can do something that's never before been possible. You can put an Altair under your tree or under the tree of a friend. Or you can start with our special Christmas 75 time payment plan!*

The Altair 8800 is the **NUMBER ONE** hobby computer in the whole world. No other computer offers you the power and versatility—the proven track record—of the Altair 8800 at a comparable price. No other hobby computer offers you the sophistication of Altair BASIC software or the wide variety of Altair modules and peripherals. No other hobby computer offers you the customer support of an Altair (free membership in the Altair Users Club, free access to the Altair Service Department and the Altair Software Library, and a free subscription to the Altair monthly publication, *Computer Notes*).

Order now and avoid the last minute Christmas rush!

* Christmas 75 Time Payment Plan

1K Altair for Just $68.00 a month!

The Altair time payment plans allow you to be the owner of an Altair 8800 with 1,024 bytes of memory for just $68.00 a month. Each month (for 8 months) you send in your payment and we send you part of an Altair kit until you have the complete system. The advantages of this plan are **NO interest** or financing charge, **GUARANTEED price** based on today's price, and free, **immediate membership** to the Altair Users Group including subscription to *Computer Notes*.

Our terms are cash with order, BankAmericard or Master Charge. If you send in an early payment we will make an early shipment. By the same token, a late payment will result in a late shipment. (After 60 days past due, the balance of the deal is cancelled. All payments must be made within 10 months).

Total Price: $544 (Retail price: Altair $439, Memory $97, Postage and handling $8—total $544)

HARDWARE PRICES:

Altair Computer kit with complete assembly instructions	$439
Assembled and tested Altair Computer	$621
1,024 Byte Static Memory Card	$97 kit and $139 assembled
2,048 Byte Static Memory Card	$145 kit and $195 assembled
4,096 Byte Dynamic Memory Card	$264 kit and $338 assembled
Full Parallel Interface Card	$92 kit and $114 assembled
Serial Interface Card RS232)	$119 kit and $138 assembled
Serial Interface Card (TTL or Teletype)	$124 kit and $146 assembled
Audio Cassette Record Interface	$128 kit and $174 assembled
COMTER II*	$780 kit and $920 assembled

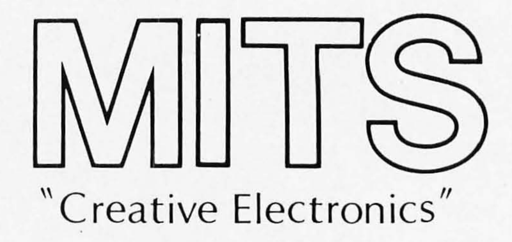
*The Comter II Computer Terminal has a full alpha-numeric keyboard and a highly readable 32-character display. It has its own internal memory of 256 characters and complete cursor control. Also has its own built-in audio cassette interface that allows you to connect the Comter II to any tape recorder for both storing data from the computer and feeding it into the computer. Requires an RS232 Interface Card.

SOFTWARE PRICES:

Altair 4K BASIC	$350
Purchasers of an Altair 8800, 4K of Altair Memory, and Altair Serial I/O or Audio-Cassette I/O	ONLY $60
Altair 8K BASIC	$500
Purchasers of an Altair 8800, 8K of Altair Memory, and Altair Serial I/O or Audio-Cassette I/O	ONLY $75
Altair EXTENDED BASIC	$750
Purchasers of an Altair 8800, 12K of Altair Memory, and Altair Serial I/O or Audio-Cassette I/O	ONLY $150
Altair PACKAGE ONE (assembler, text editor, system monitor)	
Purchasers of an Altair 8800, 8K of Altair Memory, and Altair I/O	ONLY $30

NOTE: When ordering software, specify paper tape or cassette tape.

Warranty: 90 days on parts for kits and 90 days on parts and labor for assembled units. Prices, specifications, and delivery subject to change.



MITS/6328 Linn N.E., Albuquerque, NM 87108 505/265-7553 or 262-1951

MAIL THIS COUPON TODAY!

☐ Enclosed is check for $__________
☐ BankAmericard #__________ ☐ or Master Charge #__________
☐ Altair 8800 ☐ Kit ☐ Assembled ☐ Options (List on separate sheet)
Include $8 for postage & handling
☐ Altair Time Payment Plan
☐ Please send free literature ☐ Please send my order to different address (List on separate sheet)

NAME__________
ADDRESS__________
CITY__________ STATE & ZIP__________

MITS/6328 Linn N.E., Albuquerque, NM 87108 505/265-7553 or 262-1951

NOTE: Personal checks take 2-3 weeks for clearance. For immediate processing send money order or use charge card.

THE CURVE TRACER THAT WON'T COLLECT DUST.



The Hickok Model 440 semiconductor curve tracer is all purpose and convenient to use. It's the ideal instrument for testing, evaluating, classifying and matching all types of transistors, FET's and diodes. You'll get stable, full range dynamic displays that you can accurately scale right from the screen.

- Pull-out card for easy, fast set-up and operation.
- Set-up marks for rapid set-up of 80% of tests.
- Unique INSTA-BETA display takes the guesswork out of transistor and FET parameter measurement.
- In-or-out of circuit testing.
- A full range professional tracer at a price you can afford.

AT YOUR DISTRIBUTOR **$165.00**

HICKOK

the value innovator

INSTRUMENTATION & CONTROLS DIVISION
THE HICKOK ELECTRICAL INSTRUMENT CO.
10514 Dupont Avenue • Cleveland, Ohio 44108
(216) 541-8060 • TWX: 810-421-8286

DIGITAL PERFORMANCE YOU CAN RELY ON.



The Hickok Model 334 DMM is a rugged, non-temperamental, hardworking tool that's easy to use and easy on your eyes. Hickok has established a unique reputation in digital electronics during the past 10 years. The Model 334 is another example of our engineering expertise — an economical lab quality instrument with exceptional durability and accuracy.

- Easy reading, green fluorescent display
- 3½ digit — auto polarity
- 26 ranges including 200 mV AC & DC ranges
- Fast response — 2.5 readings/sec

Basic Accuracies (% of reading)
DC Volts; ±0.2% (±0.5% on 200V, 1200V ranges)
AC Volts; ±0.5% (±2.0% on 200 mV, 2V ranges)
OHMS; ±0.5%
DC Current; ±1.5%
AC Current; ±2.0%

AT YOUR DISTRIBUTOR **$229.00**

HICKOK

the value innovator

INSTRUMENTATION & CONTROLS DIVISION
THE HICKOK ELECTRICAL INSTRUMENT CO.
10514 Dupont Avenue • Cleveland, Ohio 44108
(216) 541-8060 • TWX: 810-421-8286

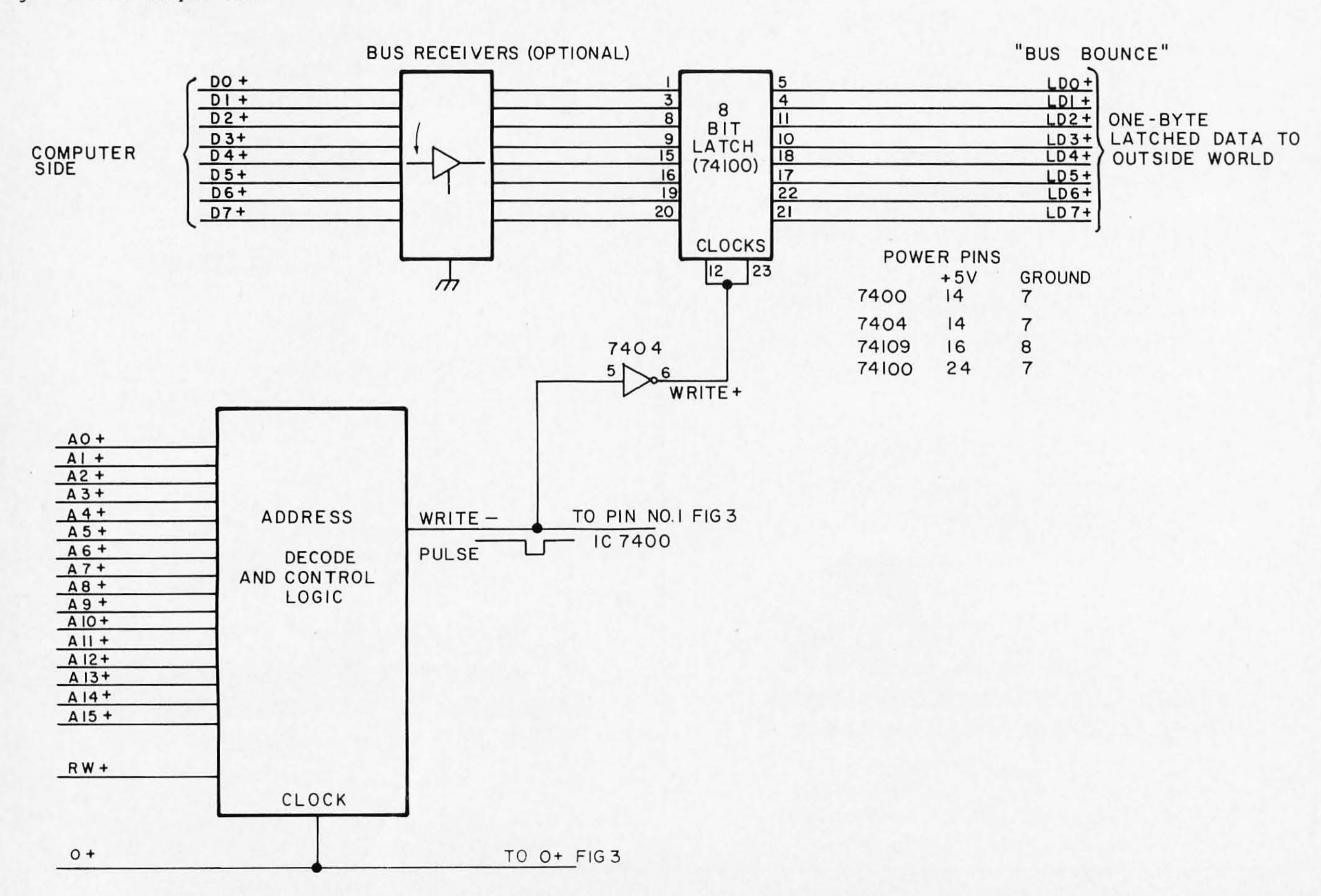
notes on parallel output interfaces

by
Carl Helmers
Editor, BYTE

One way to connect an extra output port for a teletype or other peripheral to your CPU is to make the interface simulate a memory address during the writing operations. This method is the one which is used for both the input and output functions in computers such as the PDP-11 of DEC, or the Motorola 6800 microcomputer. The method can even be used to overlap a usable main memory address since the CPU could care less whether or not the addressed port is connected in addition to the proper main memory location! The same method can even be used on computers such as the Altair 8800 which split the CPU bus into two parts and thus complicate the interface picture.

All of the microcomputers I have seen to date for the home brew computer market operate with a degree of parallelism at the bit level. Whether the chip is 4-bit, 8-bit or a 16-bitter, the concept of "parallel" data is built in. Data is parallel in nature if each bit has one line assigned to it and transfers of a group of such bits are always made simultaneously. Thus for example, the address lines used to select memory words are usually done in

Fig. 1. 8-bit bus output latch.



in memory address space

parallel in CPU designs of practical utility. With the bus oriented computer chips likely to be used for homegrown systems, it is possible to grab data from the busses by latching it in a register which listens to the bus continuously but is only written when the proper address is found. This article concerns such latching of output data and suggestions about several applications of the technique.

The basic idea of the bus output is illustrated in Fig. 1. Fig. 1 shows a set of data lines (denoted D0+ to D7+ to indicate a positive logic definition) constituting an 8-bit data bus. In a 16-bit computer, this set of lines might be one or the other half of the 16-bit data bus, or the logic might be extended to 16 bits. Connected directly to the bus pins of the interface I have noted a set of "bus receivers". This circuit should be put in if necessary to maintain consistency of bus loading with all the other bus interconnects. For instance, with a tri-state 8833 circuit as the bus definer, up to 100 high-impedance PNP-input receivers (input side of 8833) can be connected to the bus. But put a TTL load on, and the fanout will be reduced considerably. (For an Altair 8800, the data bus is split into two components: in and out. The principle of minimizing the loading of the Altair drivers (TTL) would indicate use of a low power (74Lxx) device as the bus receiver. A non-inverting receiver is to be preferred in order to keep the same logical sense of the data to be stored in the latch.)

Following the bus receiver, a latch is shown. The latch illustrated with its pinouts is the 8-bit, 24-pin package called a 74100. Alternate circuits for this function include a pair of 7475s, or even four dual master slave flip flop packages, such as 7473s. In general, it will pay to use the larger scale of integration from a power-budget standpoint. Consider the specs for two 7475s (64 mA) versus four 7473s (80 mA). For a sixteen bit output, all that is required is to double the number of bits used for the latch. The latch is used for only one purpose – to hold the data after it is stored, until updated by a later write to the same location.

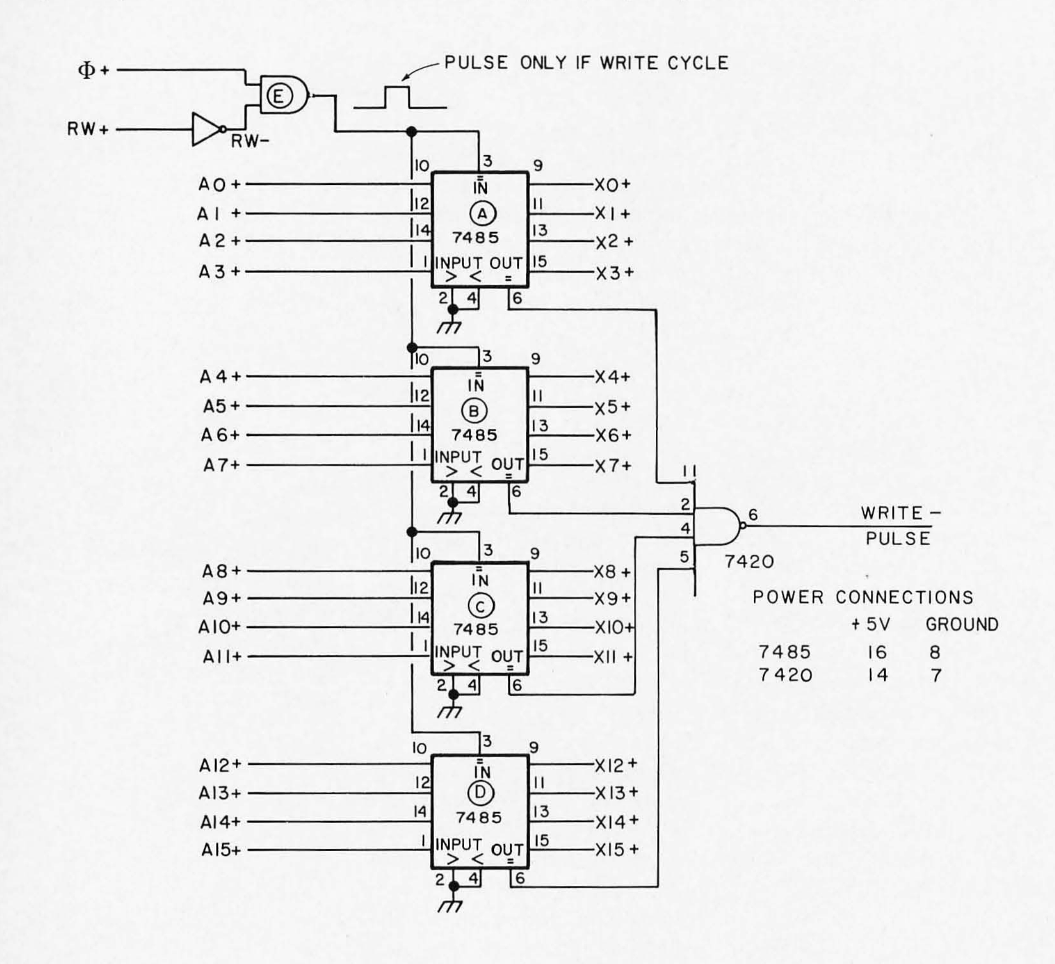
A latch is required to buffer the output logically in many instances of I/0 devices. A primary example of such a case is an output which needs stable data for a much longer period than the short CPU-cycle during which data is stable at the output of bus receivers. **If you interface your computer bus to one of the Burroughs SELF-SCAN display devices with the memory option, for instance, your data must be stable for a long period of time (about 60 microseconds). This requirement is necessitated by the need to wait for the shift register memory to cycle around to the proper position for entry of new data. If your interface is to a digital to analog converter (DAC) presenting a gain control voltage to a computerized audio mixing panel, then you would want the control signal to stay stable for all time until explicitly altered by the CPU.**

Fig. 1 is completed by the notation of a big mostly-blank box. Big blank boxes with labels in them are a way of saying "here is a function, but I haven't told you what it is in detail." In this case the function is address decode and control logic for grabbing output data. I have drawn the box with inputs indicated from 16 bits of addressing, an "RW+" signal and a "Ø+" signal. The logic of this box will respond to a specific address in order to generate a negative logic (WRITE-) pulse which is inverted and used to latch the data at the correct time. The definition of the specific address desired and the decoding are both considered a bit later when Fig. 2 is discussed. The "RW+" signal controls the direction of the CPU's data transfer. If it is logic "1" (high level) then the CPU is attempting to read data from the bus and no clock pulse is allowed to reach the latch, even if the address bits A0+ to A15+ match the desired address. If "RW+" is low, then the CPU is sending data out and a clock pulse is allowed through the address decode and control logic. The clock pulse is taken from the CPU supplied clock Ø+ and is

"Big blank boxes with labels in them are a way of saying 'here is a function, but I haven't told you what it is in detail.' "

"The method can even be used to overlap a usable main memory address since the CPU could care less . . . "

Fig. 2. Single-address 16-bit decode with 7485. "Xn" (n=Ø to 15) is logical 1 or Ø defining desire address.



a positive logic signal indicating that valid data is present.

Assuming you actually want to grab some data off the bus at a specific location when it is referenced, how can you implement the address decode and control function? Fig. 2 is a suggestion of one method to accomplish this function for a *specific location* at a considerable price in hardware: using more than one memory location defined in this way would rapidly lead to a large parts count for 7485s as decoding logic. The basic idea of Fig. 2 is to use the 7485 comparator circuits to check for equality between the address lines A0+ to A15+ and a set of "desired address" definition lines, X0+ to X15+. The comparison is done as four groups of four bits, and a parallel logical product (AND) of the results of all four bit-group comparisons is performed by the 7420. The comparators' cascading input for equality is used to enable the comparison: the AND gate "E" detects a CPU write operation as the simultaneous occurrence of the clock Ø+ and a low state of RW+.

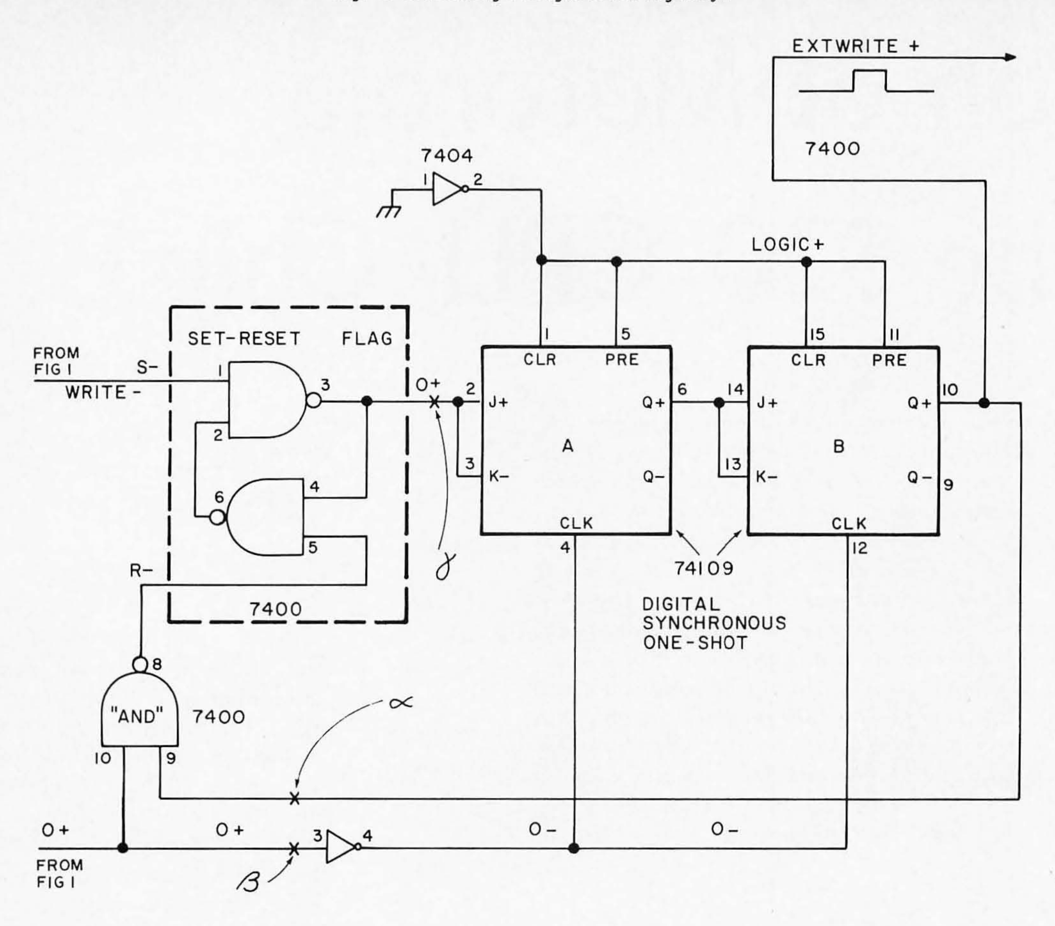
A Hardware Memory Contents Monitor

A particular application for which single-address decoding might be useful is as a debugging tool based on this circuit, used to monitor the last content written into a specific location. Such a debugging tool can be built by defining the X0+ to X15+ address lines as the outputs of a set of four hexadecimal switches or six octal encoded switches, hand set from the panel of the debugging instrument. Then the outputs of the latch circuit might be routed to a set of hex or octal LED drivers so that a display of each number written might be obtained. A more general variation of the same theme would be attainable as a bus monitor device if the gate E of g Fig. 2 is eliminated entirely and the clock Ø+ is simply used as the enable condition of the comparators (pins 3 get Ø+). Then the "memory contents monitor" always shows the contents of the memory bus at the time it was last used with the desired address.

Adding a Longer Clock

It is often necessary to obtain a clock signal which is longer than the original latching clock. In such cases the longer clock must also occur during a time when the latched data is stable, i.e., after the CPU is finished with its addressing of the output latch. One way to generate such a delayed longer clock is to use the analog timing elements called "one shots" – such as the 74122 or 74123 circuits. In order to do so, however, you will have to calculate a bunch of resistor

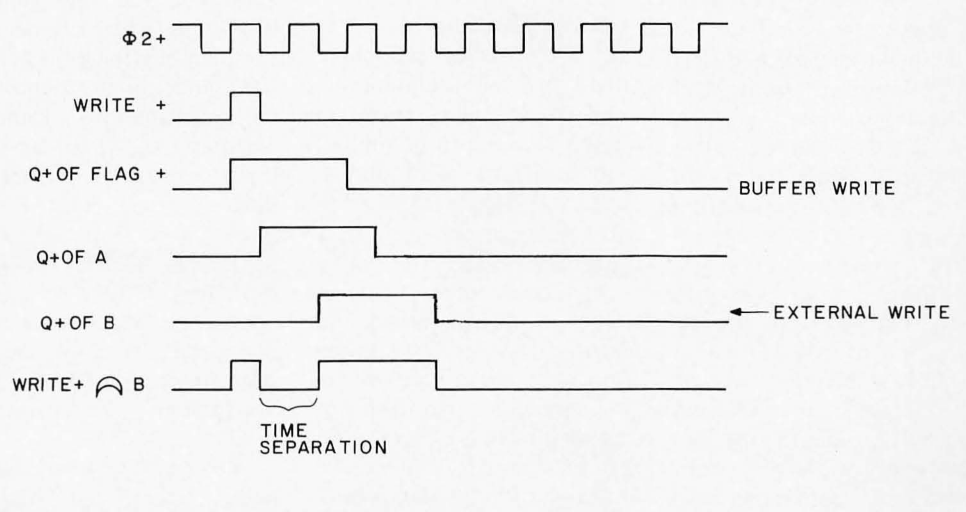
Fig. 3. Generating a longer clock digitally.



and capacitor values for the delays, put in your nearest approximations and cross your fingers. A better way to achieve a deterministic system is to use entirely "synchronous" logic concepts and delays implemented with gates and flip flops.

Fig. 3 and its corresponding timing diagram Fig. 4 is a detail of one method to cue a long but delayed clock pulse. The basic idea is to set a flag (the SR flip flop formed by the two NAND sections and labelled "flag") when the I/O write occurs. This flag becomes data which will get clocked synchronously into flip flop A, then into flip flop B. The output of flip flop B is used to enable a reset pulse to the flag, which brings the system into a stable quiescent state until the next output WRITE- pulse occurs. The timing diagram of Fig. 4 illustrates how the synchronous operation produces an auxiliary pulse (Q+ of flip flop B) which is 2 clock periods in length. This clock is delayed with respect to setting the flag by the original WRITE- pulse, but the delay is fixed and synchronous due to the fact that the actual clock (or its inverted e derivative) is used to cause all state changes of the flip flops.

Fig. 4. Timing: external write vs. buffer write.



Son of Motorola
(or, the $20 CPU Chip)

Would you believe – another microprocessor? You bet. The calculator firm, MOS Technology of Norristown, Pennsylvania, has just recently announced a new microprocessor which combines plug in compatibility with the Motorola 6800 and a new instruction set to come out with yet another option for microprocessor users – but at a price of $20 in single quantities. Here comes the under $200 processor kit? Not quite yet, but maybe within a year or two. (It's already to the point where the sheet metal and transformer iron of a home computer often cost more than all the silicon products which make it work . . . this new low on CPU prices just compounds the problem.) It may be three to six months before you see one of these new MCS6501 processors designed into a kit, so Dan Fylstra in his article covers quite a few details of the Motorola 6800 by way of comparison with "Son of Motorola."

by
Daniel Fylstra
Associate Editor, BYTE
25 Hancock St.
Somerville MA 02144

We thought that the "age of the affordable computer" had arrived when you could buy a microprocessor chip for $150. But the potent combination of new technology and free enterprise has brought about developments beyond our wildest expectations.

So now you can buy your microprocessor brand new, in single quantities, for $20. The new offering is from MOS Technology, Inc., and is pin-compatible, but software-incompatible with the Motorola 6800 microprocessor. Although it will be a while before the new chip finds its way into ready-to-build kits for the hobbyist (after all, the first Motorola 6800 kits have just been announced), the news should be of interest to nearly every home brew computer experimenter. So here's a comparison of the Motorola 6800 and the MOS Technology 6500 series, based on the information presently available. If you aren't already familiar with the Motorola microprocessor, don't worry – we'll cover its major features in the course of the comparison.

Hardware Comparison

Both the Motorola 6800 and the MOS Technology chip are TTL-compatible devices, operating from a single five volt power supply. Like earlier microcomputers, such as the Intel 8008, 8080 and National PACE, these processors make use of a bidirectional data bus, to which both memory and input/output devices may be connected. However there are no special input/output instructions in the instruction repertoire of either the Motorola or MOS Technology microprocessors. Output of a character, for example, is accomplished by storing a value into a certain memory location, which is in reality a special register inside an external I/O interface chip, connected to the data bus just like any other RAM or ROM chip.

Motorola supplies a Peripheral Interface Adapter (PIA) chip which connects to the data bus for 8-bit parallel I/O, and an Asynchronous Communications Interface Adapter (ACIA) for bit-serial input/output. (The ACIA is simply a type of UART, as discussed in Don Lancaster's September article on serial interfaces. It may be used to connect a teletype or CRT terminal to the micro-computer system.) MOS Technology plans to supply a similar set of chips.

Most of the time, data is being transmitted between the microprocessor and the memory chips over the data bus. But the processor can also disconnect itself from the bus, enabling, for example, a data transfer to take place directly between an I/O device and memory. Both the Motorola 6800 and the MOS Technology chip have three-state buffers for the eight data lines, enabling them to disconnect from the bus in this fashion. But the Motorola also has three-state buffers on its 16 address lines, whereas the MOS Technology chips do not.

This would be used, for example, in a floppy disk controller which is capable of transferring a whole block of many bytes of data in response to a single command from the CPU. The controller would present a series of addresses on the 16 address lines, and data bytes on the data lines, causing the bytes to be stored in a series of locations in some RAM chip on the bus; all this would take place in the intervals when the CPU itself was disconnected from the bus.

As a practical matter, however, small systems do not require this kind of direct memory access (DMA) capability, and larger systems with more devices on the bus will require buffers on the

Ready or not, here I come: 6800 to 6501.

address lines to supply the necessary power – and these buffers may as well have three-state outputs.

The other major hardware difference between the Motorola 6800 and the MOS Technology 6500 series is that the MOS Technology chip has an 8080-style Ready line, whereas the Motorola 6800 does not. The Ready line is used to make the microprocessor wait for a variable length of time before going on with the execution of an instruction. This feature makes it easy to use the less expensive memory chips, especially for Programmable or Erasable Read-Only Memory (PROM or EROM) which are not as fast as the CPU itself. It is possible to use such devices with the Motorola 6800, of course, by stretching out one of the clock phases to as long as five microseconds. But the availability of the Ready line on the MOS Technology chip is certainly a convenience, and allows you to use extremely slow memories if you wish.

The MCS6501, first in the MOS Technology 6500 series, requires the same type of external clock as the Motorola 6800. But for $25 you can have the MCS6502, which includes an on-the-chip clock, driven by an external single phase clock or an RC or crystal time base input. As the manufacturer suggests, it is probably cheaper in an original design to use the MCS6502 than to provide the external logic to generate the two-phase clock.

To sum up, both the Motorola 6800 and the MOS Technology have comparable features with some differences. In terms of hardware differences, the MOS Technology Ready line is probably more valuable than the three-state address line buffers available on the Motorola 6800.

A final hardware advantage possessed by the MOS Technology chip is speed. The Motorola 6800 cycle time is one microsecond (1 MHz clock rate), and a typical instruction takes about three clock cycles. While the cycle time of the MOS Technology chip is nominally the same, the company has hinted broadly that the chip can be run at clock rates of 2 or even 3 MHz. Of course, one would have to use faster and more expensive memory chips to take advantage of this increased speed.

In addition, certain critical instructions take fewer cycles on the MOS Technology chip. An STA (store accumulator) instruction referencing an arbitrary location takes 4 cycles, versus 5 for the Motorola, and a JSR (jump to subroutine) instruction requires 6 cycles, as opposed to 9 on the 6800. Conditional branches take 4 cycles on the Motorola microprocessor, while they take 2 cycles if the condition is false and 3 if it is true on the MOS Technology chip. Because these instructions are so frequently executed in most programs, the 6500 series should enjoy a performance edge over the Motorola 6800 even at the same clock rate.

Table I. Functionally equivalent instructions for both the Motorola 6800 and MOS Technology MCS6501 microprocessors. The mnemonics are Motorola's. Of course, these instructions operate on the A accumulator only in the MCS6501, but can address either accumulator in the Motorola 6800. The BIT instruction (*) has a different effect on the V and N processor flags in the MCS6501.

ADC	DEX
AND	EOR
ASL	INC
ASR	INX
BCC	JMP
BCS	JSR
BEQ	LDA
BIT*	LDX
BMI	LSR
BNE	NOP
BPL	ORA
BVC	PSH
BVS	PUL
CLC	ROL
CLI	RTI
CLV	RTS
CMP	SBC
CPX	SEC
DEC	SEI
	STA
	STX
	TSX
	TXS

Software Comparison

We can treat the instruction set architecture of the two processors in two stages, first considering the facilities for manipulating *data* and then dealing with the facilities for manipulating *addresses*. Both features are important to the overall effectiveness of the processor design.

Data Manipulation

The instructions for manipulating data are quite similar on the two processors. There are two major differences: First, the Motorola 6800 has two 8-bit accumulators, A and B, while the MOS Technology chip has only one accumulator, A. Second, in addition to conditional branches for unsigned comparisons, the Motorola 6800 has special branch instructions for signed comparisons, but the MOS Technology chip does not. (The signed comparisons treat the two values as positive or negative numbers in two's complement notation, in the range –128 to +127. For example, –1 is represented as 2^8-1 = 11111111. An unsigned comparison would treat this quantity as the largest possible (8-bit) value, whereas a signed comparison would treat it as smaller than, say, zero.)

We thought that the "age of the affordable computer" had arrived when you could buy a microprocessor chip for $150. But the potent combination of new technology and free enterprise has brought about developments beyond our wildest expectations.

Table I lists the instructions which are the same for both processors, while Table II lists instructions on the Motorola 6800 which must be replaced by more than one instruction on the 6500 series microprocessors.

Some of the instructions omitted on the MOS Technology chip are merely incidental; others are more serious. The lack of signed comparisons represents a real inconvenience in many applications. The lack of a simple ADD instruction means that an operation such as A = B + C on one-byte operands must be coded with a "Clear Carry" (CLC) as in this example:

```
CLC
LDA  B
ADC  C
STA  A
```

on the MOS Technology chip. On the other hand, a computation such as A = B + C – D could be coded as

```
CLC
LDA  B
ADC  C
SBC  D
STA  A
```

assuming that the inclusion of "carry" in both operations is indeed desired.

Less serious but still irritating are the absence of the ROR (rotate right), NEG (negate) and COM

Table II. Motorola 6800 instructions which have no direct equivalent in the MCS6501. The information in this table is taken from MOS Technology documentation on the 6500 series.

Motorola 6800 Instruction	Equivalent 6500 Series Sequence
ABA	No B accumulator
ADD	CLC, ADC
BGE loc	BMI *+6, BVC loc, BVS *+4, BVS loc
BGT loc	BMI *+6, BVC *+6, BVS *+6, BVC *+4, BNE loc
BHI loc	BCS *+4, BNE loc
BLE loc	BEQ loc, BMI *+6, BVS loc, BVC *+4, BVC loc
BLS loc	BCS loc, BEQ loc
BLT loc	BMI *+6, BVS loc, BVC *+4, BVC loc
BRA	JMP
BSR	JSR
CBA	No B accumulator
CLR [loc]	LDA #0, [STA loc]
COM [loc]	[LDA loc], EOR #$FF, [STA loc]
DAA	Replaced by SED
DES	Use PHA
INS	Use PLA
LDS loc	LDX loc, TXS
NEG [loc]	EOR #$FF, ADC #1 [or LDA #0, SBC loc]
ROR [loc]	[LDA loc], PHP, LSR, PLP, BCC *+4, ORA #$80, [STA loc]
SBA	No B accumulator
SEV	LDA #1, LSR
STS loc	TSX, STX loc
SUB	CLC, SBC
SWI	BRK saves state without transferring control
TAB	No B accumulator
TAP	PHA, PLP
TBA	No B accumulator
TPA	PHP, PLA
TST	BIT #0
WAI	JMP *
op disp, X [indexed addressing mode]	LDY #disp, op @loc, Y [indirect indexed addressing mode]

(complement) instructions, as well as single-byte instructions to increment and decrement the accumulator. Probably the least significant difference is the omission of the B accumulator on the MOS Technology chip. This is more than made up for by the availability of an extra index register (see below).

All in all, the Motorola 6800 comes out ahead when considering facilities for manipulating data, the most important point in its favor being the availability of the signed comparisons. Generally speaking, however, the basic instructions available on the two processors are quite similar.

Address Manipulation

The greatest architectural differences between the two processors lie in their facilities for manipulating addresses, or their "addressing modes" — and here the MOS Technology chip has much more to offer.

The two microprocessors are the same in one respect: both have special "short forms" of most instructions for referencing the first 256 bytes of memory. This is called "direct addressing" on the Motorola 6800, and "zero page addressing" on the MOS Technology chip. As an example, the most general LDA (load accumulator) instruction is three bytes long; the second and third bytes form the effective address (0-65535), which can reference any byte in memory. The short form of the LDA instruction, however, is two bytes long; the second byte forms the effective address (0-255) of a byte in the first "page" of memory. The "short form" instructions generally take one fewer clock cycle to execute, since only two rather than three instruction bytes must be fetched from memory.

The major differences between the two processors lie in the important area of indexed addressing. The Motorola 6800 has a single 16-bit index register, called X. Essentially all instructions have an indexed addressing form, in which a one-byte displacement (0-255) is added to the address in the index register to form the effective address. The MOS Technology chip, on the other hand, has two 8-bit index registers, called X and Y. All of the computational instructions have indexed addressing forms in which either a one- or two-byte base address is added to the contents of either the X or the Y register to form the effective address.

Which approach is the better one? For the purpose of accessing elements of arrays, or tables of many identical elements, the MOS Technology chip comes out way ahead. This is partly due to the lack of certain critical instructions on the Motorola 6800, such as an instruction to add the contents of an accumulator to the index register, or even to transfer the value in the accumulators to the index register.

Suppose that we wish to add the Ith element of an array, S_I, to another variable, T. In general, the array may be located anywhere in memory, and the subscript I may be the result of some calculation done in the accumulators. Letting S denote the address of the zeroth element (the base address) of the array, and assuming that the value of the subscript I is already in the A accumulator, consider the instructions necessary to accomplish this operation on the two processors.

The biggest difference is in the area of addressing modes, an area where the 6500 series devices far outshine the Motorola 6800.

On the Motorola 6800, our first try yields the following:

```
SHI  EQU S/256*256
     CLR B          ]
     ADD A #S-SHI   ]  Calculate the
     ADC B #S/256   ]   indexed address
     STA A TEMP+1   ]
     STA B TEMP     ]
     LDX TEMP       ]
     LDA A 0, X     ]  Perform desired
     ADD A T        ]   computation
     STA A T        ]
```

This instruction sequence requires 19 bytes, counting the two-byte temporary TEMP and assuming that TEMP and T are located in the first 256 bytes of memory. Since the array S could be anywhere in memory, we were unable to use the displacement field of an instruction with indexed addressing for the array base address, and instead we had to add the array base to the index (in double precision), store the result in memory, load it into the index register, and finally reference the array element S_I.

We can improve on this with the aid of a little lateral thinking. Noticing that the 6800 is actually capable of adding a one-byte quantity to a two-byte address, but only in a storage reference with indexed addressing, we will split up the base address into two parts to arrive at a better solution:

```
SHI    EQU S/256*256
       STA A TEMP+1
       LDX TEMP
       LDA A S-SHI, X
       ADD A T
       STA A T
       .
       .
       .
TEMP   FDB SHI
```

This instruction sequence requires only 12 bytes, under the same assumptions.

Even so, we can't match the simplicity of the solution to the same problem on the MOS Technology chip:

```
TAX
LDA S, X
ADD T
STA T
```

This instruction sequence requires only seven bytes. Only four bytes were needed to reference the element S_I, versus eight for the Motorola 6800.

How important is this improvement? It is certainly significant, since arrays and tables are used so frequently in programs of any size. On the other hand, in many applications it is only necessary to reference each element of an array in turn; it is not necessary to access elements randomly based on a computed subscript. In this case, we can obtain better code on the Motorola 6800 by first loading the array base address into the index register, and then referencing each element directly (i.e., with a zero indexed address displacement), incrementing the address in the index register using the INX instruction to proceed from element to element. We are therefore using the 6800's index register to hold a pointer or indirect address rather than an index.

An even more important difference between the two microprocessors in that the MOS Technology chip possesses *two* (8-bit) index registers, X and Y, whereas the Motorola 6800 has only *one* (16-bit) index register X. As we shall see, two index registers are far more valuable than two accumulators. This is because programs frequently manipulate two (or more) tables, or other indirectly addressed variables, at the same time. As an example, we will consider perhaps the simplest operation of this type, the problem of moving a string of bytes from one area of storage to another. Assume that 20 bytes, starting at the location denoted by the symbol FROM, are to be moved to the area starting at the location denoted by the symbol TO.

Two index registers are far more valuable than two accumulators.

On the Motorola 6800, we can write the following routine:

```
LOOP   LDX FRPTR ]  Fetch
       LDA A O, X ] FROM
       LDX TOPTR ]  Move
       STA A O, X ] TO
       INC FRPTR ]  change
       INC TOPTR ]  pointers
       DEC COUNT ]
       BNE LOOP  Test
                 continuation
       .
       .
       .
FRPTR  FDB FROM
TOPTR  FDB TO
COUNT  FCB 20
```

This routine requires 24 bytes, including the working storage locations, and executes in 820 clock cycles. This routine can move up to 256 bytes.

On the MOS Technology chip we have the following solution:

```
       LDX #0
       LDY #0
LOOP   LDA FROM, X
       STA TO, Y
       INX
       INY
       DEC COUNT
       BNE LOOP
       .
       .
       .
COUNT  FCB 20
```

This routine requires 17 bytes, and executes in 404 clock cycles. The improvement in speed clearly depends on the number of bytes to be moved; each pass through the loop in the Motorola 6800 routine takes 41 clock cycles, while each pass through the loop in the MOS Technology routine takes 20 cycles. (The MOS Technology routine is also limited to moving at most 256 bytes.)

Once again the degree of improvement is substantial, and the improvement is

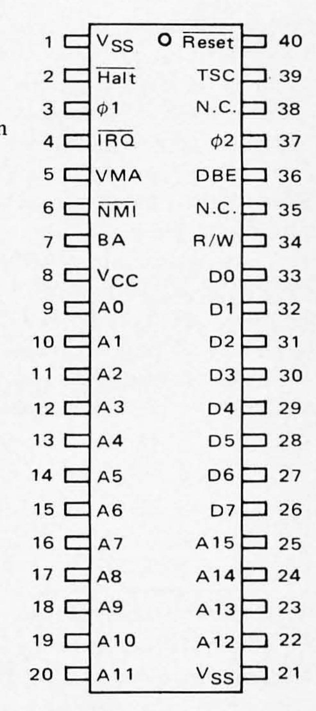


Fig. 1. The pin assignments of the Motorola 6800 (and by implication, the MOS Technology MCS6501). V_{SS} is ground (0 volts) and V_{CC} is +5 volts. The A lines are address outputs, and the D lines are bidirectional tristate data bus lines. For details see the Motorola and MOS Technology documentation of these parts.

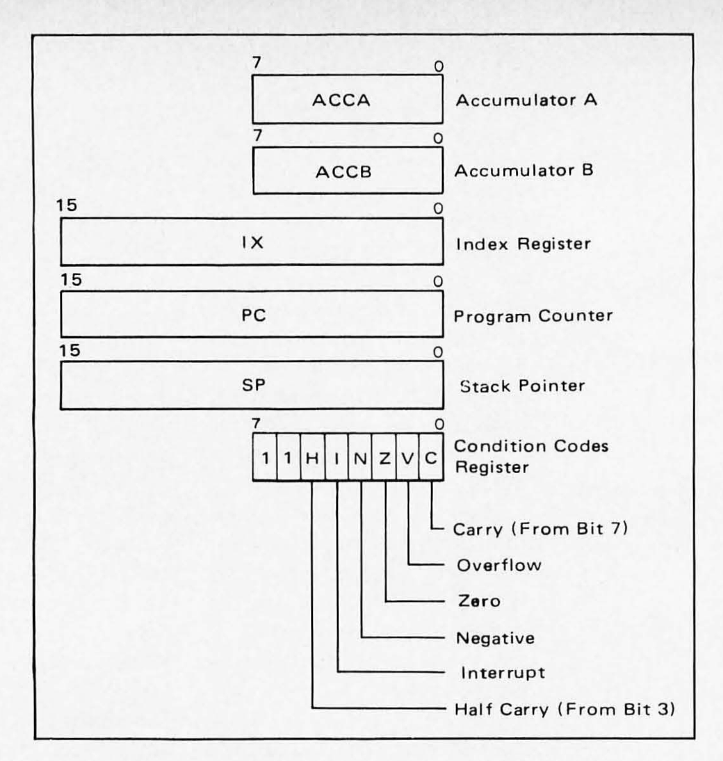


Fig. 2. The programmer's view of the 6800 CPU. This diagram, excerpted from the Motorola 6800 documentation, shows the various registers of the CPU including the processor's condition code register. Note the similarity to the MCS6501 in Fig. 3.

significant because this type of problem arises so frequently in large programs.

The MOS Technology chip has some additional addressing modes not possessed by the Motorola 6800. First, there is a "short form" for instructions with simple indexed addressing if the array base address is in the first "page" (256 locations) of memory. This feature is of somewhat limited use except in very small programs, since only a few small arrays can actually be placed in the first 256 locations. Of greater interest is the so-called "indirect indexed" addressing mode. Instructions with this type of addressing are two bytes long; the second byte specifies the address of a two-byte constant in the first page of memory. This two-byte constant then becomes the "array base address," and the contents of the Y register are added to this constant to form the effective address. This addressing mode is very useful: In a program with many references to a particular array or table which is too large to place in the first page of memory, one can *trade space for time* by placing the array base address in the first page of memory, and then referencing elements of the array using indirect indexed addressing. Each element reference takes less space (two bytes instead of three) but more time (five cycles instead of four) than would be required for ordinary indexed addressing.

One unfortunate feature of the MOS Technology chip's many addressing modes is that they do not apply consistently to all instructions.

There are two other addressing modes on the MOS Technology chip which are somewhat less useful. The first is called "indexed indirect" addressing: Here the contents of the X register are added to a one-byte base address to obtain the address of a two-byte constant in the first page of memory. The contents of this two-byte constant then becomes the effective address. Unfortunately this addressing mode is not available for the JMP instruction, where it would be most useful: It could be used to implement a "jump table," or a "computed GO TO" or "CASE statement" in some high-level languages.

Finally, two other addressing modes are used with branch instructions:

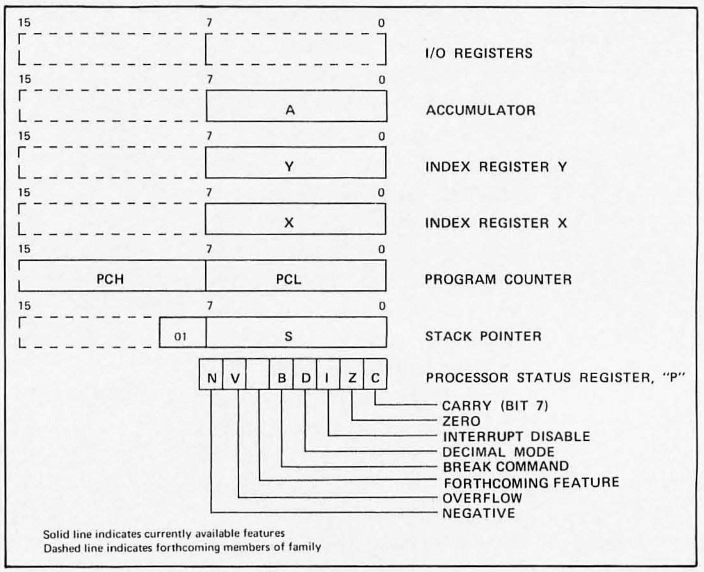


Fig. 3. The programmer's view of the MCS6501 CPU. This diagram, excerpted from the MOS Technology 6500 series preliminary documentation, shows the various registers of the CPU. Note the similarity to the Motorola 6800 diagram in Fig. 2.

Table III. Instructions, addressing modes and execution times for the Motorola 6800 processor. Execution times are in "machine cycles" which for a 1.0 MHz clock take 1.0 microsecond apiece. This table is excerpted from Motorola documentation on their processor.

	(Dual Operand)	ACCX	Immediate	Direct	Extended	Indexed	Implied	Relative
ABA		•	•	•	•	•	2	•
ADC	x	•	2	3	4	5	•	•
ADD	x	•	2	3	4	5	•	•
AND	x	•	2	3	4	5	•	•
ASL		2	•	•	6	7	•	•
ASR		2	•	•	6	7	•	•
BCC		•	•	•	•	•	•	4
BCS		•	•	•	•	•	•	4
BEA		•	•	•	•	•	•	4
BGE		•	•	•	•	•	•	4
BGT		•	•	•	•	•	•	4
BHI		•	•	•	•	•	•	4
BIT	x	•	2	3	4	5	•	•
BLE		•	•	•	•	•	•	4
BLS		•	•	•	•	•	•	4
BLT		•	•	•	•	•	•	4
BMI		•	•	•	•	•	•	4
BNE		•	•	•	•	•	•	4
BPL		•	•	•	•	•	•	4
BRA		•	•	•	•	•	•	4
BSR		•	•	•	•	•	•	8
BVC		•	•	•	•	•	•	4
BVS		•	•	•	•	•	•	4
CBA		•	•	•	•	•	2	•
CLC		•	•	•	•	•	2	•
CLI		•	•	•	•	•	2	•
CLR		2	•	•	6	7	•	•
CLV		•	•	•	•	•	2	•
CMP	x	•	2	3	4	5	•	•
COM		2	•	•	6	7	•	•
CPX		•	3	4	5	6	•	•
DAA		•	•	•	•	•	2	•
DEC		2	•	•	6	7	•	•
DES		•	•	•	•	•	4	•
DEX		•	•	•	•	•	4	•
EOR	x	•	2	3	4	5	•	•

	(Dual Operand)	ACCX	Immediate	Direct	Extended	Indexed	Implied
INC		2	•	•	6	7	•
INS		•	•	•	•	•	4
INX		•	•	•	•	•	4
JMP		•	•	•	3	4	•
JSR		•	•	•	9	8	•
LDA	x	•	2	3	4	5	•
LDS		•	3	4	5	6	•
LDX		•	3	4	5	6	•
LSR		2	•	•	6	7	•
NEG		2	•	•	6	7	•
NOP		•	•	•	•	•	2
ORA	x	•	2	3	4	5	•
PSH		•	•	•	•	•	4
PUL		•	•	•	•	•	4
ROL		2	•	•	6	7	•
ROR		2	•	•	6	7	•
RTI		•	•	•	•	•	10
RTS		•	•	•	•	•	5
SBA		•	•	•	•	•	2
SBC	x	•	2	3	4	5	•
SEC		•	•	•	•	•	2
SEI		•	•	•	•	•	2
SEV		•	•	•	•	•	2
STA	x	•	•	4	5	6	•
STS		•	•	5	6	7	•
STX		•	•	5	6	7	•
SUB	x	•	2	3	4	5	•
SWI		•	•	•	•	•	12
TAB		•	•	•	•	•	2
TAP		•	•	•	•	•	2
TBA		•	•	•	•	•	2
TPA		•	•	•	•	•	2
TST		2	•	•	6	7	•
TSX		•	•	•	•	•	4
TSX		•	•	•	•	•	4
WAI		•	•	•	•	•	9

NOTE: Interrupt time is 12 cycles from the end of the instruction being executed, except following a WAI instruction. Then it is 4 cycles.

Which processor comes out ahead overall? To a great extent it depends on your point of view: Systems programs are better on the MOS Technology machines; applications programs would tend to come out ahead on the Motorola 6800.

"Relative" addressing, available on both the Motorola and the MOS Technology processors, is used with the conditional branch instructions, which are two bytes long. The second byte of such an instruction specifies a positive or negative displacement in two's complement notation (−128 to +127). The destination address of the branch is taken to be the algebraic sum of the address of the byte immediately following the branch instruction and this displacement. Of course, this means that it is possible to branch directly to a location within only a certain limited distance from the branch itself; but, more often than not, the range of −128 to +127 bytes is adequate, and a space savings is realized in comparison to processors such as the Intel 8080 which have only three-byte branch instructions. If necessary, a conditional branch can always transfer to a three-byte unconditional JMP instruction, which can jump to any location in memory. On the MOS Technology chip, a JMP instruction can also specify "absolute indirect" addressing: In this case, the second and third bytes of the instruction specify the address of a two-byte constant anywhere in memory, and the contents of this two-byte constant becomes the destination address for the jump.

One unfortunate feature of the MOS Technology chip's many addressing modes is that they do not apply consistently to all instructions. For example, the binary arithmetic instructions are available with essentially all addressing modes, but the unary arithmetic instructions are missing the Y-register and indirect modes, and the BIT instruction is missing several others as well. This not only makes programming more difficult, since one must constantly check to see which instruction forms are legal, and program around the exceptions; it also makes the design of an assembler or compiler more complicated. A compiler, in particular, would require complex logic to determine when it could and could not take advantage of the addressing modes.

In summary, the MOS Technology chip comes out ahead when considering facilities to manipulate addresses, and in many cases the advantage realized due to the availability of the extra addressing modes is substantial. The greatest failing of the 6500 series design is the inconsistent availability of the addressing modes from instruction to instruction.

Which processor comes out ahead overall? This is very difficult to judge. It depends partly on whether the programs being executed on the microcomputer are "system" programs, such as compilers, interpreters and I/O controllers, which tend to make heavy use of address

Table IV. Instructions, addressing modes and execution times for the MOS Technology MCS6501 processor. Execution times are in "machine cycles" which for a 1.0 MHz clock take 1.0 microsecond apiece. This table is excerpted from MOS Technology documentation on their processor.

	Accumulator	Immediate	Zero Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Implied	Relative	(Indirect, X)	(Indirect), Y	Absolute Indirect
ADC	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
AND	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
ASL	2	.	5	6	.	6	7	.	.	.	.	.	.
BCC	.	.	.	.	.	.	.	.	.	2**	.	.	.
BCS	.	.	.	.	.	.	.	.	.	2**	.	.	.
BEQ	.	.	.	.	.	.	.	.	.	2**	.	.	.
BIT	.	.	3	.	.	4	.	.	.	.	.	.	.
BMI	.	.	.	.	.	.	.	.	.	2**	.	.	.
BNE	.	.	.	.	.	.	.	.	.	2**	.	.	.
BPL	.	.	.	.	.	.	.	.	.	2**	.	.	.
BRK	.	.	.	.	.	.	.	.	.	.	.	.	.
BVC	.	.	.	.	.	.	.	.	.	2**	.	.	.
BVS	.	.	.	.	.	.	.	.	.	2**	.	.	.
CLC	.	.	.	.	.	.	.	.	2	.	.	.	.
CLD	.	.	.	.	.	.	.	.	2	.	.	.	.
CLI	.	.	.	.	.	.	.	.	2	.	.	.	.
CLV	.	.	.	.	.	.	.	.	2	.	.	.	.
CMP	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
CPX	.	2	3	.	.	4	.	.	.	.	.	.	.
CPY	.	2	3	.	.	4	.	.	.	.	.	.	.
DEC	.	.	5	6	.	6	7	.	.	.	.	.	.
DEX	.	.	.	.	.	.	.	.	2	.	.	.	.
DEY	.	.	.	.	.	.	.	.	2	.	.	.	.
EØR	.	2	3	4	.	4	4*	4*	.	.	6	5	.
INC	.	.	5	6	.	6	7	.	.	.	.	.	.
INX	.	.	.	.	.	.	.	.	2	.	.	.	.
INY	.	.	.	.	.	.	.	.	2	.	.	.	.
JMP	.	.	.	.	.	3	.	.	.	.	.	.	5

	Accumulator	Immediate	Zero Page	Zero Page, X	Zero Page, Y	Absolute	Absolute, X	Absolute, Y	Implied	Relative	(Indirect, X)	(Indirect),Y	Absolute Indirect
JSR	.	.	.	.	.	6	.	.	.	.	.	.	.
LDA	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
LDX	.	2	3	.	4	4	.	4*	.	.	.	.	.
LDY	.	2	3	4	.	4	4*	.	.	.	.	.	.
LSR	2	.	5	6	.	6	7	.	.	.	.	.	.
NØP	.	.	.	.	.	.	.	.	2	.	.	.	.
ØRA	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
PHA	.	.	.	.	.	.	.	.	3	.	.	.	.
PHP	.	.	.	.	.	.	.	.	3	.	.	.	.
PLA	.	.	.	.	.	.	.	.	4	.	.	.	.
PLP	.	.	.	.	.	.	.	.	4	.	.	.	.
RØL	2	.	5	6	.	6	7	.	.	.	.	.	.
RTI	.	.	.	.	.	.	.	.	6	.	.	.	.
RTS	.	.	.	.	.	.	.	.	6	.	.	.	.
SBC	.	2	3	4	.	4	4*	4*	.	.	6	5*	.
SEC	.	.	.	.	.	.	.	.	2	.	.	.	.
SED	.	.	.	.	.	.	.	.	2	.	.	.	.
SEI	.	.	.	.	.	.	.	.	2	.	.	.	.
STA	.	.	3	4	.	4	5	5	.	.	6	6	.
STX	.	.	3	.	4	4	.	.	.	.	.	.	.
STY	.	.	3	4	.	4	.	.	.	.	.	.	.
TAX	.	.	.	.	.	.	.	.	2	.	.	.	.
TAY	.	.	.	.	.	.	.	.	2	.	.	.	.
TSX	.	.	.	.	.	.	.	.	2	.	.	.	.
TXA	.	.	.	.	.	.	.	.	2	.	.	.	.
TXS	.	.	.	.	.	.	.	.	2	.	.	.	.
TYA	.	.	.	.	.	.	.	.	2	.	.	.	.

* Add one cycle if indexing across page boundary
** Add one cycle if branch is taken, Add one additional if branching operation crosses page boundary

In favor of the 6500 series are price and speed; in favor of the 6800 are availability and very good Motorola documentation.

manipulation facilities; or application programs, which make greater use of data manipulation facilities. One would expect better results in the former case with the MOS Technology chip, and in the latter case with the Motorola 6800. One would also expect the MOS Technology chip to enjoy an advantage on large programs, since larger programs inevitably tend to make use of tables, subroutines with parameters, and other forms of address manipulation.

All in all, the Motorola 6800 comes out ahead when considering facilities for manipulating data . . . but nevertheless the two processors are quite similar.

Against these factors one must weigh the availability of an excellent applications manual, proven software, and kits for the hobbyist for the Motorola 6800 microprocessor. At the same time, the MOS Technology chip's price can't be beat, and its speed advantage may be important for some purposes.

At the time that this article is being written (late August), the MOS Technology chip is just a promise: The chip should be available for purchase at the Western Electronics Conference (Wescon) in San Francisco, September 16-19. By the time you read this, the chip itself should be in the hands of at least a few hobbyists. Let's have some letters to BYTE describing initial experiences with the new microprocessor! Send your comments to the author or to the editor of BYTE. In the meantime, we'll be waiting to see what new surprises the semiconductor houses and kit manufacturers have in store for us. And BYTE will try to keep you up to date on the latest developments in the world's hottest, fastest-moving hobby – home computers!

Table V. MCS6501 microprocessor instructions, listed in alphabetical order by mnemonics. The instructions with asterisks are similar to the same mnemonics in the Motorola 6800 processor.

*ADC	Add with Carry to Accumulator	*JSR	Jump to New Location Saving Return Address
*AND	"AND" to Accumulator	*LDA	Transfer Memory to Accumulator
*ASL	Shift Left One Bit (Memory or Accumulator)	*LDX	Transfer Memory to Index X
		LDY	Transfer Memory to Index Y
*BCC	Branch on Carry Clear	*LSR	Shift One Bit Right (Memory or Accumulator)
*BCS	Branch on Carry Set		
*BEQ	Branch on Zero Result	NØP	Do Nothing - No Operation
*BIT	Test Bits in Memory with Accumulator	*ØRA	"OR" Memory with Accumulator
*BMI	Branch on Result Minus	*PHA	Push Accumulator on Stack
*BNE	Branch on Result not Zero	PHP	Push Processor Status on Stack
*BPL	Branch on Result Plus	*PLA	Pull Accumulator from Stack
*BRK	Force an Interrupt or Break	PLP	Pull Processor Status from Stack
*BVC	Branch on Overflow Clear	*RØL	Rotate One Bit Left (Memory or Accumulator)
*BVS	Branch on Overflow Set		
*CLC	Clear Carry Flag	*RTI	Return From Interrupt
CLD	Clear Decimal Mode	*RTS	Return From Subroutine
*CLI	Clear Interrupt Disable Bit	*SBC	Subtract Memory and Carry from Accumulator
*CLV	Clear Overflow Flag	*SEC	Set Carry Flag
*CMP	Compare Memory and Accumulator	SED	Set Decimal Mode
*CPX	Compare Memory and Index X	*SEI	Set Interrupt Disable Status
CPY	Compare Memory and Index Y	*STA	Store Accumulator in Memory
*DEC	Decrement Memory by One	*STX	Store Index X in Memory
*DEX	Decrement Index X by One	STY	Store Index Y in Memory
DEY	Decrement Index Y by One	TAX	Transfer Accumulator to Index X
*EØR	Exclusive-or Memory with Accumulator	TAY	Transfer Accumulator to Index Y
*INC	Increment Memory by One	*TSX	Transfer Stack Register to Index X
*INX	Increment X by One	TXA	Transfer Index X to Accumulator
INY	Increment Y by One	*TXS	Transfer Index X to Stack Register
*JMP	Jump to New Location	TYA	Transfer Index Y to Accumulator

More information on the 6500 series microprocessors is available from:

MOS Technology, Inc.
Valley Forge Corporate Center
950 Rittenhouse Rd.
Norristown PA 19401
1-215-666-7950

Information on the Motorola 6800 microprocessor is available from many local distributors, and from:

Motorola Semiconductor Products Inc.
Box 20912
Phoenix AZ 85036

GLOSSARY

BYTE's Board of Resident Inexperts (BRI) has ruled the following terms to be worthy of further explanation. This list is probably not complete – readers who would like further explanation of terminology are invited to write a letter to the editor identifying terms which need such treatment.

8-Bit Bidirectional Bus – a "data bus" which simultaneously transmits eight separate signals corresponding to one byte's worth of information. The bidirectional aspect means that either tristate, open collector or similar form of output stage is used, so that multiple drivers can be tied in common with only one such driver active at any time. A given board, CPU, output terminal or other logic circuit can then interface to the bus (with some addressing and master timing control intelligence) for both sending and receiving data.

Effective Address – whenever the computer's CPU addresses memory, it must send out 16 bits (for Motorola 6800, MCS 6501 or other similar chips). The way in which these 16 bits are derived can often be a fairly elaborate procedure, as well as a simple absolute expression. Whatever the method of derivation, however, the result is a 16-bit value which is used to address memory, called the effective address because it is what actually does go out to memory regardless of the details of the internal codes of the program.

Instruction Repertoire – the repertoire of a musician is the set of all pieces he or she can play well in concert. Well, the repertoire of a computer – its instructions – is the list of all the instructions it can perform and their definitions.

Subscript – in typical high order languages, a means is provided to specify elements of arrays of data. This is done by subscripts to indicate the "nth" element for subscript "n". Use of such notation presents the problem of calculating the *effective address* of the actual data being referenced. In the context of evaluating a CPU, attention spent on the problem of calculating effective addresses from subscripts is very fundamental.

Time Base – whenever it is necessary to examine the relative timing of different signals, it is necessary to have a reference point and a scale for making the measurement. This is the "time base" of the reference.

TTL compatible – one of the largest families of integrated circuits is the line of "transistor-transistor logic" devices, TTL for short. A TTL compatible line of some non-TTL device can "drive" one or more TTL loads if it is an output, or can receive a TTL device's output if it is an input. There are various cautions to be observed – probably worthy of a BYTE article – when different types of logic are interfaced, but the phrase "TTL compatible" usually means that the compatible device can be wired directly to TTL interconnection pins safely in at least one configuration.

Unary – this term is derived from the Latin roots of "oneness." A unary operation is an operation which has but one operand, for example the complement operation of a Motorola 6800 CPU.

Monitor 8½ —

Your Own Pseudo Instructions

Monitor 8 is a program which was written by people at the now defunct Microsystems International semiconductor operation. The program was published in the MF8008 Applications Manual, *1974 Edition, Bulletin 80007, by Microsystems International, Ltd., Box 3529 Station C, Ottawa, Canada, K1Y 4J1. I'm working at the problem of getting permission to reproduce the copyrighted materials in sections C (User's Guide) and D (Software Listing) for the benefit of the 8008 hackers in BYTE's readership. Monitor 8 is a "systems program" designed to make life easier for you by performing various little "utility" functions like editing octal data in memory, loading and dumping to cassette tape, copying data from place to place in memory, translating machine codes to mnemonics, setting and clearing break points, etc. The people who have the applications manual of MI and use Monitor 8 are usually enthusiastic about it . . . witness the following set of comments sent in by Willard I. Nico concerning use and extensions of the program as it was printed in the original manual.*

by
Willard I. Nico
Delta t
11020 Old Katy Road, Suite 204
Houston TX 77043

I've fallen in love with "Monitor 8". The noble effort by Programmer Tom of Microsystems International has made it easier and more fun to write, edit and manipulate programs. But the Monitor can be even more fun, more enjoyable and more useful if you get inside its little head and stir things up a bit.

Roll Your Own Pseudo Instruction!

Say you've got a routine starting at memory location 012000 that turns on the coffee pot at 6:00 am. Before you go to bed you type in XQT 012000 and the program is running. That's great, but if you have a lot of routines, you have to look up the proper starting address if you forget. How about just entering COF and let Monitor 8 find it for you?

What you have done is assign a mnemonic to your routine and called it with a pseudo instruction, PI for short.

What's a Pseudo Instruction?

The 8008 CPU recognizes 48 basic instructions. Counting the variations, there are 228 in all. In mnemonic form, LAB is an instruction recognized by the CPU. But you can LAC, LAD, LAE and so on, so the *basic* instruction is Lr_1r_2 and the different combinations are variations of that basic instruction. (r_1 or r_2 can be any one of the mnemonics A, B, C, D, E, L, H or M.)

If you come up with a three-letter combination, such as DLP, that is not one of the variations recognized by the CPU, but which will cause something useful to happen, it's called a pseudo instruction. When you are running the Monitor 8 program, DLP will cause the current location pointer to be printed on your output device, and therefore it is a PI.

So how about COF? Or TKL (if you have the proper output device), or EAT, SLP, RUN? They can all be PIs if the desired program runs when you type in the mnemonic. The only restriction is that your mnemonic can't be the same as one that is recognized by the CPU. For example, JMP would be a no-no.

How Do I Implement my Own Pseudo Instructions?

I thought you would never ask!

The first thing to do is relocate the Five Byte Table that now lives at 004021 through 004155. Since we are going to add our PIs to this table, we have to get some room and it's now hemmed in by the DPS Output routine and the Three Byte Table. I suggest you start it on a fresh page of memory. Once you start adding your own PIs you will get carried away, so leave lots of room at the end of the Table.

(The Five Byte Table is the pseudo instruction command table for Monitor 8. This table contains 5-byte groups consisting of three bytes of pseudo instruction name followed by a 2-byte pseudo instruction service routine address. There is one 5-byte group for each pseudo instruction that Monitor 8 understands. There is a psuedo instruction service routine to execute each different pseudo instruction.)

Let's assume that you choose 007000 as the starting point for your new Five Byte Table. With the Monitor 8 running, type

```
CPY
004021
004155
007000
```

and your handy COPY Routine will move the Table for you.

Now that the Table has been moved, we need to change the reference to its

location in the main program. That's easy too, thanks to Monitor 8. Type

EDT
*
003153
000
(space)
(space)
007

and your new Table is in use.

Because of automatic addressing between the Three Byte Table and Five Byte Table for the HLT, RST, INP, OUT and ??? symbolic dumps, a portion of the Five Byte Table (004127 through 004155) will have to be left at the old location as well as being part of the Table at the new location. The rest of the old addresses (004021 through 004126) can be used to store any new routines you can fit in.

Add a Pseudo Instruction

Now we are ready to add a new PI to the Monitor 8 repertoire. Let's take the COF program at 012000. We add the mnemonic to the Five Byte Table, followed by the lower and then the higher addresses. I think the easiest way is with the EDT routine. Look up the octal equivalents for the letters C, 0 and F. We will add the PI to the Five Byte Table starting where the ??? is and then replace the ??? at the end of the Table. So type

EDT
*
007132
103 "C" (ASCII)
117 "O"
106 "F"
000 } address of COF routine
012 }
077 "?"
077 "?"
077 "?"

Now that we have a new Table entry, we need to extend the number of entries the Five Byte Table search routine will go through before giving up and calling it a "no find".

The original Table had 022 (octal) listings, so we simply change it to 023.

EDT
*
003151
023

will take care of it nicely.

Getting Fancy

You can implement PIs for routines that need a starting and ending address just as easily. Just remember to add "200" to the high address before recording it in the Table. As an example, suppose you want to add RUN and the routine needs to know where to run from and to. If the RUN program resides at 013000, your new Table entry would be

122
125
116
000
213

When the PI is typed in, the Monitor will respond with a * to ask for a starting and ending address. Your RUN program can then include a call to 001023 for data at the first and each succeeding address and a call to 000362 each time around to increment the current location pointer; compare it to the desired ending address and return to the Monitor when finished.

So there you have it. A new dimension for the fine Monitor 8 program. Give it a try and I know you will like it. HPY PGM.

Gremlin Chasing the Monitor 8

The hours spent re-coding and punching the Microsystems International "Monitor 8" software into my system have been exceptionally rewarding. To put it mildly: It's great! To Programmer Tom, wherever you are, thanks for an outstanding effort.

While using the Monitor, I found a confounding gremlin popping up inexplicably in my otherwise operational programs. Several times I tracked malfunctions down to a code that bore no resemblance to the one I had originally put there.

Sound familiar? Take heart! I have tracked the gremlin to his lair and herewith formulate the liberal dose of gremlinocide that will put him belly-up.

The problem begins with the Clear Breakpoint (CBP) routine beginning at 001353. This routine is addressed by the CBP pseudo instruction and is also called by the Set Breakpoint (SBP) routine.

Memory locations 013365, 013366 and 013367 are used as a stash for the original instruction, low address and high address when a breakpoint is set. Before setting a new one, or when clearing the breakpoint, the instruction is replaced at the address stashed.

The Clear Breakpoint routine sets the H and L pointers at the original instruction (013365) stash and retrieves it to the E register. It then increments the L pointer to the location of the high address stash (013367) and calls the "Set H and L" routine, entering at 001027.

Theoretically, the high address is 100 to indicate a clear breakpoint. This 100 is tested by loading the D register with 100 at 001365 and exiting at 001370 if a match indicates that the breakpoint is already clear.

The problem is that the "Set H and L" routine exits with the *current instruction* in the A register, not the H address as the programmer anticipated. When the comparison is made with the 100 previously loaded into the D register, there will never be a match (unless the current instruction is JFC) and the current instruction will be replaced with whatever is at 013365. You can verify this by noting the LAM instruction at 001034 *after* the H and L pointers have been set.

Since the random codes occurring in the memory elements at the stash location on power-up are usually high numbers, the instruction mostly gets changed in a non-existent location; but not always. That's what made it such a once-in-a-great-while goof.

Of the several ways to fix the problem, I found one that does not require any added programming steps and can be implemented nicely.

Change 001365 to load the D register with "065". Change 001370 to RFZ. Now when the *instruction* at the location addressed by the H and L stash is tested, nothing will change unless the instruction is Breakpoint Execute call RST 060 (code 065).

Note that instructions 001372 through 001376 can be NOP'd since they are "do nothings." —w.i.n.

A Versatile Read Only Memory Programmer

by
Peter H. Helmers
Box 6297, River Station
Rochester NY 14611

Volatile or non-volatile? That is the question. Should your software be set in logical concrete, or should it be input from a mass storage device such as audio tape every time you start up the system? For certain small and frequently used utility software routines, dedicating a portion of your computer's memory address space to PROM is a useful technique. Peter Helmers supplies this article on a means to overcome the most difficult hurdle of the technique – programming the PROMs themselves.

The prime candidates for PROM memory are routines to extend your microcomputer's limited instruction set in software, and that fundamental utility program – the bootstrap loader. For example, on the next home brew computer I'll wrap up, I have in mind a 63-byte ROM loader program which will be blown into two 8223 ROMs using this programmer. Then when I push "load" to restart the system I'll automatically go to a loop which reads in the "real" software off an audio tape cassette, then branches to the entry point of the software on completion of the load. Some of the microcomputer kit manufacturers have begun to deliver system software in ROMs using the larger mask programmable and UV-erasable PROMs. Even if you build a kit rather than a home brew system design, you may find this programmer useful when adding extra subroutines to the existing software to customize your computer.

The read only memory, or ROM, is a versatile element in digital electronics. In one case, I've been able to replace over three packages of integrated circuits with a single ROM. In addition to this, circuits designed around ROMs tend to be less cluttered; the ROM virtually forces a cleaner design. ROMs can work wonders.

For the amateur, though, ROMs are not always easy to use. Mask programmed ROMs are programmed by the manufacturer and tend to carry high setup charges. Programmable read only memories, or PROMs, are available from electronics distributors such as Hamilton-Avnet or Cramer. These distributors will also program the PROMs that they sell. However, for the small user, the virgin PROMs not only cost more than surplus PROMs, but most distributors will also charge a one time setup charge of from $5 to $25.

The current surplus price for PROMs is about one third of the current distributor price (for an 8223 the figures are $3 surplus vs. $8.95 new). Thus the amateur whose resources are limited can save the most money by buying surplus PROMs and then programming them himself.

This article describes a circuit for a PROM programming unit for some commonly available surplus PROMs; the PROMs which can be programmed are the 8223 (32 word by 8 bits), the Schottky versions of the 8223: the 82S23 and 82S123, and the 82S26 and 82S29 (256 words by 4 bits).

The programmer programs any of these PROMs one bit at a time in strict accordance to Signetics recommended specifications. It is also capable of verifying and/or examining any bit of these PROMs to ensure that they have been programmed properly (or that "surplus" doesn't mean "preprogrammed" – but then I've received "new" PROMs from distributors that have been preprogrammed . . .). From my experience, it takes about an hour to manually program a PROM, and then verify each bit. However, to avoid mistakes, take time and work away from disturbances.

Programming Concepts

To program a given bit in a PROM, the word address is set up on some address switches, and the bit within the addressed word is selected, using an output selector switch. As shown in Fig. 1, the circuitry involved in addressing is set up in a general manner to allow addressing of either of the two PROM organizations (e.g., either 256 x 4 or 32 x 8).

Due to lack of foresight by the PROM designers, there are three sets of programming procedures which are needed for the five types of PROMs which this programmer was designed for. However, conceptually, they all require control of three PROM parameters: the PROM voltage (ROMVcc), fusing current (I_f) and $\overline{CE}$ logic level. What varies is the value of these voltages and currents, and the order in which they are applied.

The timing can be represented in general as shown in Fig. 2. The effect of each control signal will be discussed separately for each of the programming procedures.

8223 Programming Procedure

K1 when active corresponds to grounding (applying a logic "0") the $\overline{CE}$

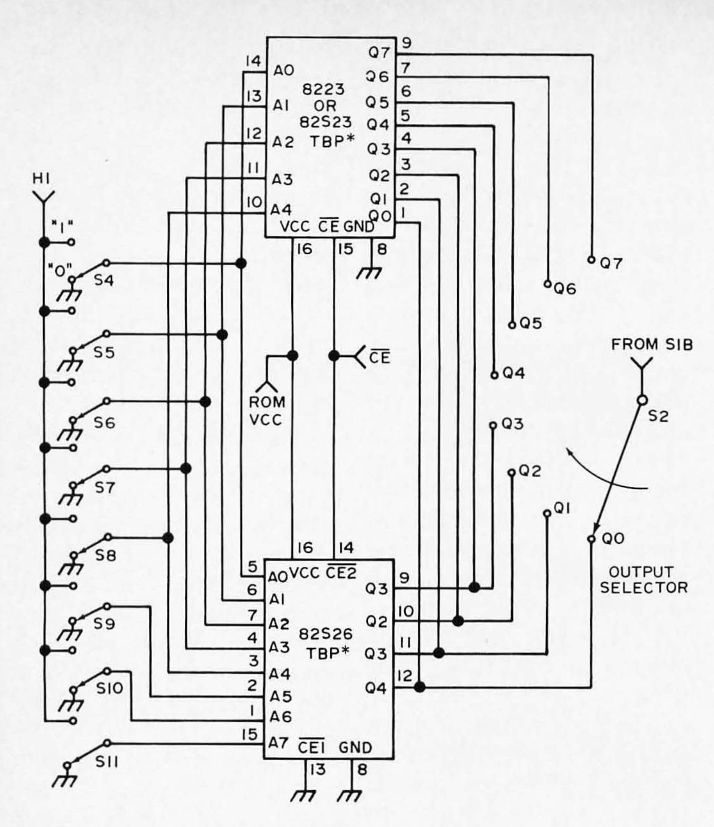


*Fig. 1. Addressing and data input configuration for a manually controlled version of the programmer. *To be programmed.*

For software design, use of an ROM can replace volatile RAM memory for routines you always want to have around.

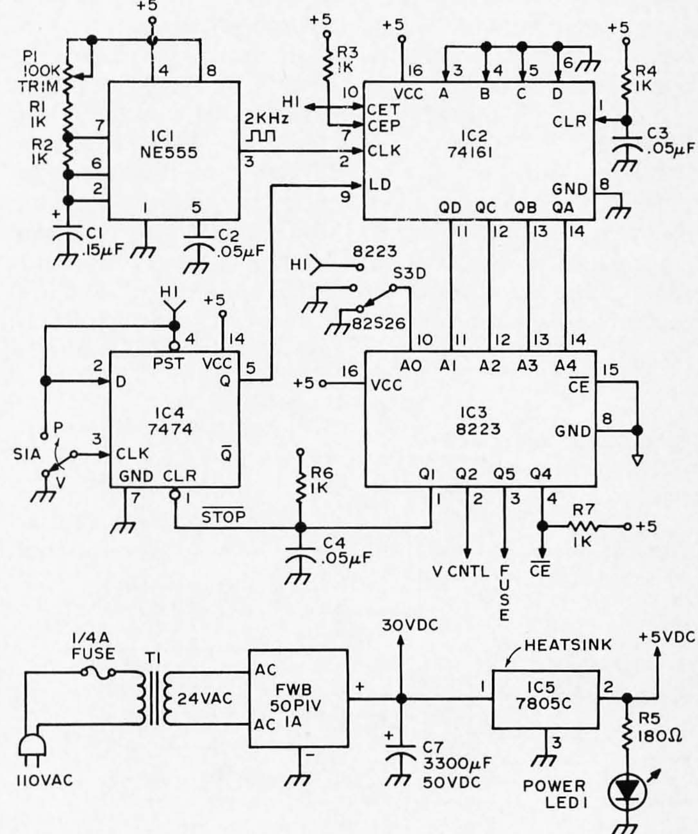


Fig. 3. Timing sequencer. This is the logic used to implement the timing diagram of Fig. 4 with the program of Fig. 5.

input of the 8223. Next, when K2 is active, a fusing current is fed into the selected PROM output. This current is specified by Signetics as the current resulting from 12.5 volts applied through a 390Ω resistor to the PROM output. The final step, when K3 is active, is to raise the PROM's Vcc (pin 16) to 12.5 volts. (The circuits that control these voltages will be discussed later.)

82S23/82S123 Programming Procedure

K1 when active raises the 82S23's Vcc to 10 volts. Then, when K2 is active, a fusing current of 65 mA is applied to the selected output of the PROM. Finally, during K3, the PROM's $\overline{CE}$ input is is grounded.

82S26/82S29 Programming Procedure

The procedure for the 82S26 and 82S29 PROMs is identical to that for the 82S23 and 82S123 except that the Vcc is raised to 12.5 volts, and the fusing current is 89 mA.

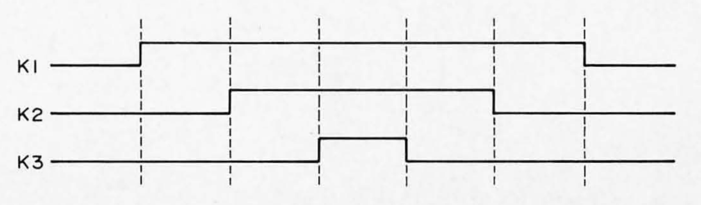
Circuit Design

Basically, there are three parts to this circuit which are shown in Figs. 1, 3 and 4. Fig. 3 shows the timing sequencer which has the responsibility of developing the proper control signals required for both programming and verifying. The control is accomplished by using an 8223 PROM IC3 in connection with a state counter. (This brings up the question of "which came first: the programmer or the PROM?") Referring to Fig. 5, the programmer starts out in a verify state upon powering up or after programming a bit. In this state, the 74161 state counter IC2 is cleared to zero and the counter enable flip flop, 7474 IC4, is cleared. Despite the fact that the 555 oscillator is producing a 2 kHz square wave, the counter is inhibited from counting because its load input is low.

At this time, as shown in Fig. 5, the PROM's $\overline{CE}$ input is held low so that the selected output may be examined by means of the "OUTPUT" indicator LED lamp.

A programming sequence is initiated by pushing the P/V push-button switch momentarily into the program (P) position. This causes the counter enable flip flop to be clocked high, allowing the state counter to

Fig. 2. "Kontrol" signals K1, K2 and K3 – relative timing.



count. The PROM IC3 controls what happens in each state. States 1 through 4 are "do-nothing" states to allow the P/V switch to stop bouncing. These states address words 00001 through 00100 or words 10001 through 10100 of PROM IC3 depending upon the A0 input at pin 10. States 5 through 14 are then used for the programming procedure's execution.

What happens during states 5 to 14 depends upon the type of PROM being programmed, as was discussed above. Since the *sequence* of events is the same for the 82S23 and 82S26 types of PROMs it can be implemented in common. Thus for these PROMs, the A0 input of IC3 is tied low so that the Vcntl, FUSE, and $\overline{CE}$ control lines will be sequenced as shown in Fig. 5. In other words, the A1 through A4 inputs of IC3 are sequencing through the program in PROM storage at words 00101 through 01110.

For the 8223 PROM, the A0 input of IC3 is high so that the control lines are sequenced by the alternate program stored in words 10101 through 11110 of IC3.

However, words 01111 and 11111 of IC3 are the same so that the following action is taken in state 15 (regardless of type of PROM). In this state, the $\overline{STOP}$ output of IC3 is low so that the counter enable flip flop is cleared, and the state counter is reset to state 0 on the next clock pulse. Since the load line is held low, the state counter will stay in this state until IC4 is clocked high when another bit is to be programmed.

Fig. 4 shows the voltage and current sources that generate the programming signals detailed above. There are two aspects to the control of these sources: type of PROM, and the control signals from the state sequencer.

The basic voltage source is shown in Fig. 6. This circuit can be modified by using a transistor switch to selectively short out the zener diode. Thus when the zener is shorted, the output voltage will be 5 volts; when the zener is not shorted, the output will be the sum of the zener voltage and the regulator's normal 5 volt output. Referring to Fig. 4(a), the transistor switch is contained in IC7a. Thus when Vcntl is low, the transistor conducts so that the value of ROMVcc is 5 volts. When Vcntl is high the transistor doesn't conduct so that the ROMVcc voltage is increased to either 10 volts or 12.5 volts depending on which zener is switched into the circuit by S3c (which depends on the type of PROM to be programmed).

The basic current source is shown in Fig. 7. Note that a zener diode is required to limit the voltage developed at the ground pin of the regulator. This diode is used to protect the PROM being programmed. Because the zener protection voltage and the value of the fusing current varies for the different PROMs, switches S3a and S3b select the V_p and I_f values required. But the 8223 requires a current defined by 12.5 volts supplied through a 390 Ohm resistor. Thus switches S3a and S3b also switch circuit into the voltage source configuration discussed above. Switch S3e then selects the fusing current from either the output of the current source circuit configuration, or from the 390 Ohm resistor required for the 8223.

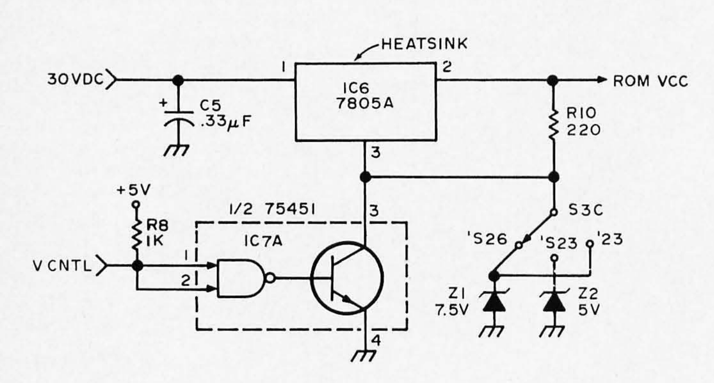
IC7b, which is controlled by the FUSE signal, is used to route the fusing current. If FUSE is low, the transistor in IC7b will conduct so that no current is applied to the selected PROM output. When FUSE is high, the transistor won't conduct so that programming can occur.

Fig. 1 shows the addressing circuitry for the programmer and requires no further comment.

Construction

The prototype (which was designed and constructed in a period of about 28 hours) was built using wire wrap techniques. All discrete components were soldered

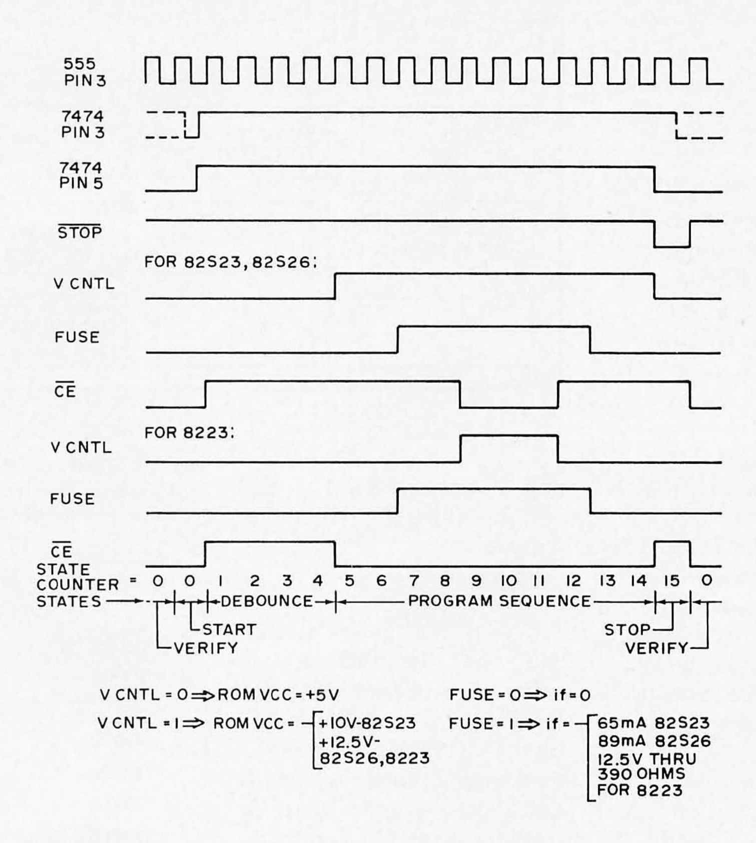
Fig. 4. Voltage and current sources.



a. Voltage source for ROMVcc.

b. Current source for I_f routed to PROM outputs via S2.

Fig. 5. Timing diagram of the sequencer. This is a synchronous timing generator illustrating one use of ROM logic. The relative timing is built into the ROM program of 8223 IC3 – with a master control of the rate set by the clock of 555 IC1.



into DIP headers so that wiring could be done to wire wrap sockets for these parts as well. The board layout that was used is shown in Fig. 8. There are two items which should be noted. Be sure to use heatsinks for the voltage regulator ICs since they have to dissipate a lot of power (due to the use of a single 30 volt unregulated supply).

The front panel of the programmer was arranged as shown in Fig. 9. Thus most of the switch wiring was done on the front panel, and interconnection to the wire wrapped circuit board was accomplished by using two plugs made from DIP headers. About 12 inches of wire was used for these interconnects. It is also advisable to include several ground wires to decrease noise.

Note that this design requires the use of a preprogrammed PROM (e.g., IC3: an 8223). This PROM must be programmed according to the truth table of Fig. 10. Probably the easiest way to get this PROM is to take advantage of the offers of the various surplus PROM sellers which advertise in BYTE and other magazines, and have them program this "bootstrap" PROM. Their prices tend to be reasonable – especially since this is the last time you'll have to pay to have a PROM programmed.

Before programming your first PROM, a single adjustment is required. This entails setting potentiometer P1 so that the NE555's clock frequency is 2 kHz. However, this setting is not critical so that if you don't have any means of measuring frequency, just set the pot near its maximum resistance setting (so that a frequency $<$ 2 kHz is obtained).

Operation

1. *Turn on:* Place the PROM in the proper socket. Plug the programmer into a 110 volt socket (note that capacitors C3 and C4 are used to initialize the programmer to state 0 so that no spurious bits will be programmed). Observe that the POWER LED is lit up.

2. *Address Setup:* The programmer verifies and/or programs only one bit of a PROM at a time. The word address is set by switches labeled A0 through A7 (but only switches A0 through A4 are used for the 32x8 PROMs; switches A5 through A7 are not electrically connected to the socket for these PROMs). The bit within the addressed word is selected by the output selector switch. (But outputs Q4 through Q7 are not electrically connected for the 256x4 PROMs).

3. *Verify/Program:* By default (of de operator ...?), the LED labeled OUTPUT indicates the status of the selected output of the addressed word. This LED is on if the bit is a logical "1" (e.g.,

Fig. 6. Voltage pedestal technique using a 5 V regulator IC.

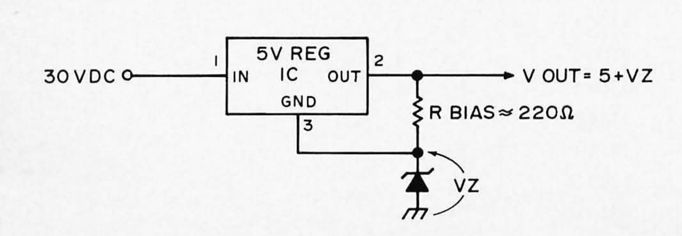
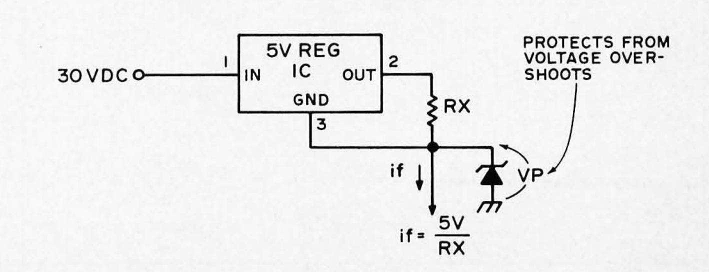


Fig. 7. Current source implemented with a 5 V regulator IC.



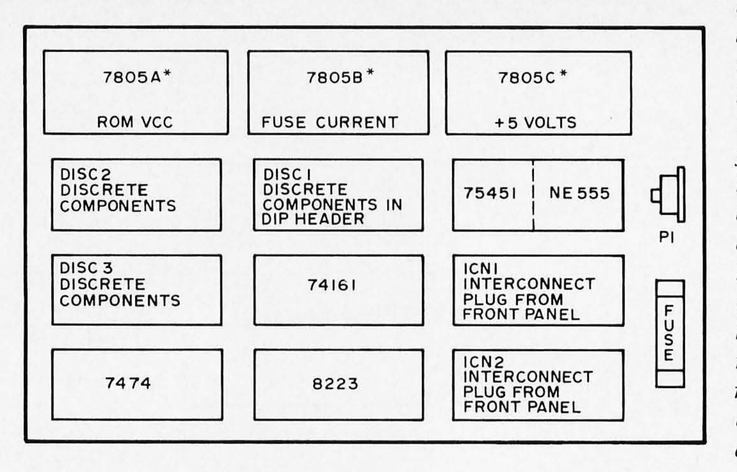


*Fig. 8. Wire wrap board layout. *Mounted on heatsinks.*

ROMs are not the "universal" logic elements that some people tend to consider them. Used correctly in logic designs, they can benefit a given design; used improperly they can cause never ending grievances.

>~2.4 volts), and is off if the bit is a logical "0" (e.g., <~.8 volts). Since the PROMs for this programmer come (hopefully . . .) with "0"s in every bit, programming involves selective fusing of "1"s in given bits. The fusing process is accomplished by pressing the PROGRAM/VERIFY switch towards the PROGRAM position, and then releasing the switch. The OUTPUT LED should then be on, to verify that the bit has been programmed. If it doesn't fuse, a second attempt can be made. However PROMs have a reputation of less than perfect fabrication – one brand new PROM that I encountered required a fusing current duration of close to 4 seconds for one output; I now use that PROM to test out new PROM programmer designs.

Final Thoughts

ROMs are great devices, and now the amateur can easily experiment with them. However my experience has been that ROMs are not the "universal" logic elements that some people tend to consider them. Used correctly in logic designs, they can benefit a given design; used improperly they can cause never ending grievances.

As a means to turn software into "firmware" the only problem with an ROM technique is making sure your program is debugged before you blow those irreplaceable fusible links inside the PROM package.

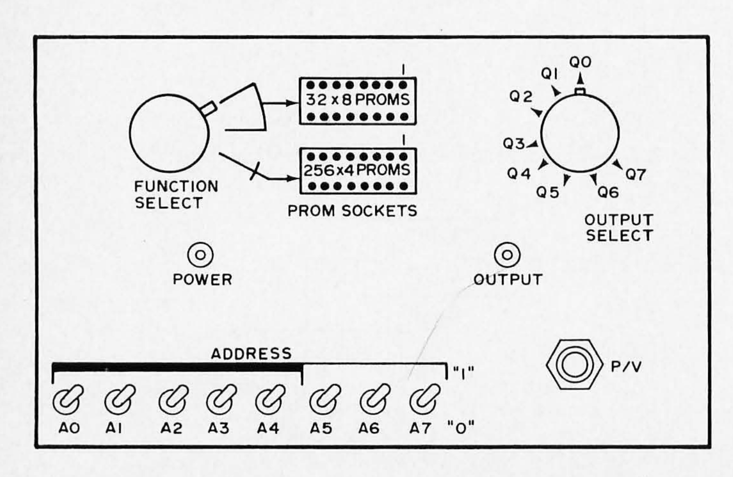
Extensions and Modifications

The PROM programmer as originally designed and described in this article is set up for a fairly tedious hand-operated technique. This technique is appropriate for occasional use, but is hardly acceptable for extensive programming. An extension which could easily be made is to replace the manual switches S1b, S2 and S3 with banks of surplus reed relays driven from TTL gates, replace S1a with a clock pulse derived from your computer, and replace the address switches with the outputs of

Fig. 10. Table of PROM bits to be "blown" to implement the program in IC3. A similar table should be made for each PROM which is to be programmed for general use once the programmer has been built.

ADDRESS	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Ao A4	STOP	Vcntl		CE	FUSE			
00000	1			1				
00001	1							
00000	1							
00001	1			1				
00010	1			1				
00011	1			1				
00100	1			1				
00101	1	1		1				
00110	1	1		1				
00111	1	1		1	1			
01000	1	1		1	1			
01001	1	1			1			
01010	1	1			1			
01011	1	1			1			
01100	1	1		1	1			
01101	1	1		1				
01110	1	1		1				
01111				1				
10000	1							
10001	1			1				
10010	1			1				
10011	1			1				
10100	1			1				
10101	1							
10110	1							
10111	1				1			
11000	1				1			
11001	1	1			1			
11010	1	1			1			
11011	1	1			1			
11100	1				1			
11101	1							
11110	1							
11111				1				

Fig. 9. Front panel layout.

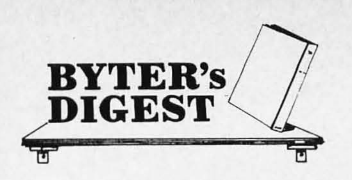


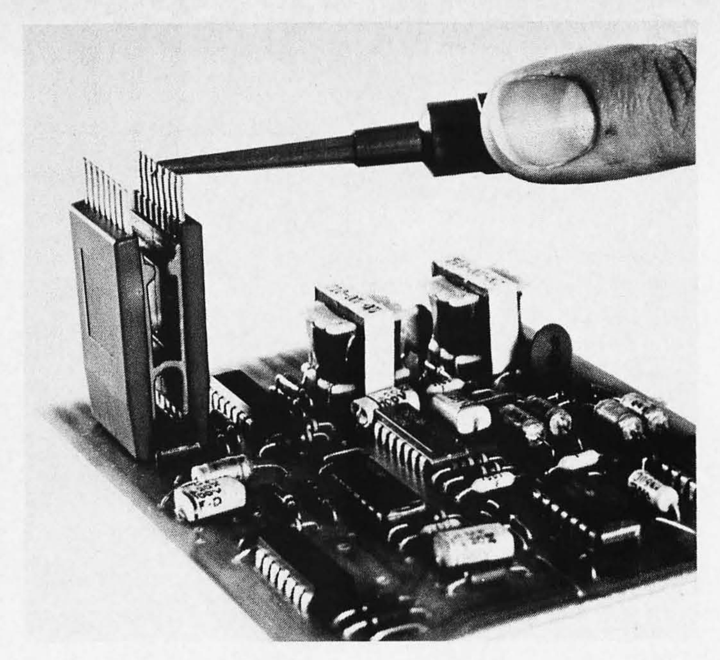
an 8-bit latch set by the computer's data bus. As inputs to the computer you'd supply the logic 1/logic 0 verify information provided by the LED indicator as well as a "programmer busy" signal derived from the "Counter Enable" line of IC4 pin 5. Put it all together with software to move data from main memory to the PROM and verify the results automatically, and you'll have a project to extend this design. Such a PROM programmer would be a good idea as a local computer club project since burning PROM programs is likely to be of common interest to many members.

PARTS LIST

R1, R2, 1k ¼ W 10%
R3, R4, 1k ¼ W 10%
R6, R7, 1k ¼ W 10%
R8, R9, 1k ¼ W 10%
R5, R14 180Ω ¼ W 10%
R10, R15 220Ω ¼ W 10%
R11 56Ω ¼ W 10%
R12 150Ω ¼ W 10%
R13 160Ω ¼ W 5%
R16 390Ω ¼ W 10%
P1 100k trim pot
C1 .15 uF
C2, C3, C4 .05 uF
C5, C6 .33 uF
C7 3300 mF 50 V electrolytic
IC1 NE555
IC2 74161
IC3 8223 (programmed as per Fig. 5)
IC4 7474
IC5, IC6, IC8 780S (5 V regulator – TO220 case)
IC7 75451
LED1, LED2 any common LED
S1 DPDT push-button
S2 SP8T rotary
S3 5P3T rotary
S4→S11 SPDT toggle
Z1, Z5 7.5 V zener diode 1N4737 1 W
Z2 5 V, 1 W zener 1N4733
Z3 22 V, 1 W zener 1N4748
Z4 16 V, 1 W zener 1N4745
T1 24 V ac transformer, 300 mA
FWB full wave bridge rectifier 50 piv, 1 Amp

Misc:
¼ A fuse and holder
line cord
heatsink for 7805s
chassis





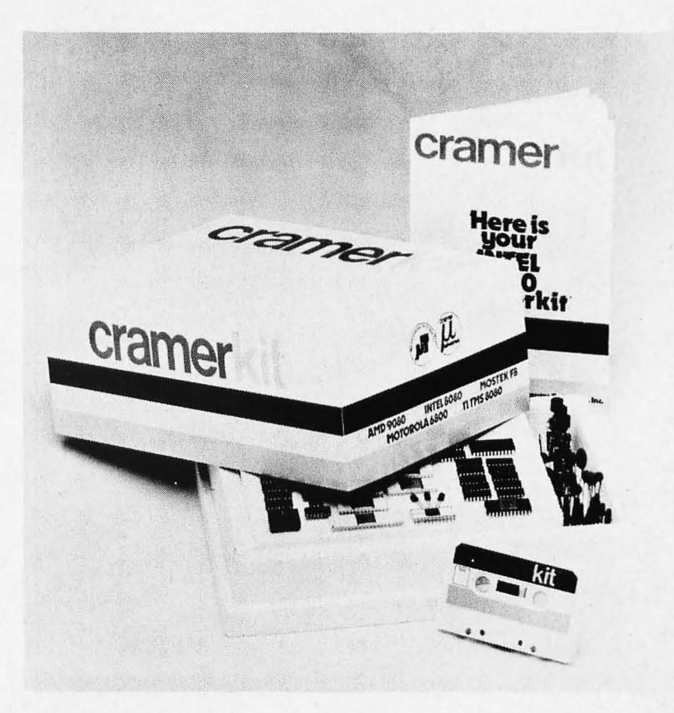
You Wouldn't Want To Be Without One of These

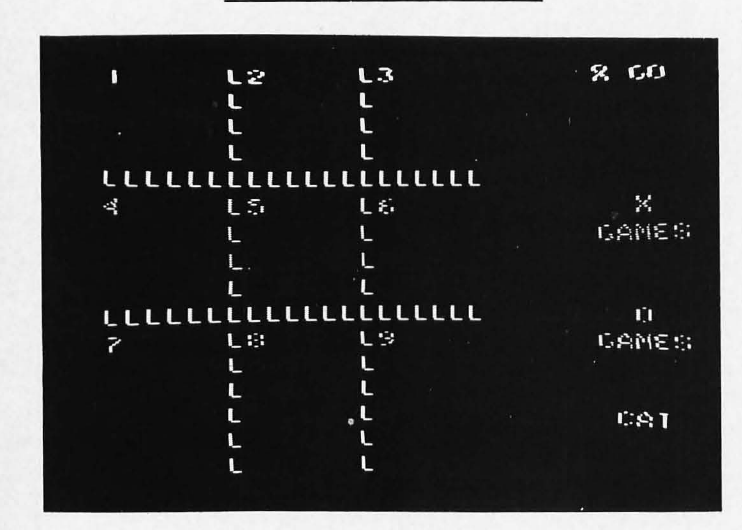
Did you ever wonder how to tap the pins of an integrated circuit mounted on a PC board – but without shorting two pins or losing grip with the clips? This photo illustrates a very useful hardware debugging tool for kit builders – Continental Specialties Corporation's "Proto Clip" in operation grabbing an integrated circuit and providing test points for probes. According to their latest catalog brochure this clip sells for $4.75 in a 16-pin DIP version.

The tool works best with ICs that have been soldered into a PC board as is often the case with both kit and manufactured products. The test probe can be attached to the extended pins for "live" monitoring of signals. The same people make a variation called "Proto Clip Cables" which start at $7.50 and come with lengths of ribbon cable pre-attached. For full details write Continental Specialties Corp., 44 Kendall St., New Haven CT 06512 – or use the reader service page of BYTE.

What Distributors Can Do for the Microcomputer Hacker

One source of the parts you can use in making your own home brew microcomputer system is the traditional electronics wholesaler. An example of one such wholesaler's product which may be of interest to some BYTE readers is the recently announced line of Cramerkit products. These products, distributed by Cramer Electronics Inc., main office 85 Wells Ave., Newton MA 02159, are packages of semiconductors and documentation which were designed and prepared by Microcomputer Technique Inc. under the direction of Jerry Ogden. I called C. L. Grant, manager of marketing services for Cramer, after he sent me a package of materials on the kits in late August. While Cramer is not really aiming the package at the amateur market, I did confirm that Cramer will sell the kits to individuals who order it and present $495. They are not intending to give out quantity orders of this package.





What To Do With a TV Display . . .

Here is one example of what to do with a TV typewriter style display hooked up to a computer. Jim Fry, PO Box 6585, Toledo OH 43612, sends us this picture taken of the face of his TV, driven by a TV-typewriter/8008 CPU combination. The display is a representation (using ASCII characters) of a simple program bringing to mind a game of Tic-Tac-Toe. The program is doing what computers do best – acting as a bookkeeper for two players sitting at the keyboard. A bit of artificial intelligence work is required to produce the classical computer which plays against the human opponent – a much more complicated program. The numbers of the game's positions are used for inputting the information on moves – the players simply type a number from 1 to 9 for an unoccupied square. This picture illustrates the board at the start of a game.

The kits come in versions oriented towards the different microprocessors for which Cramer has a franchise. The presently available kits are centered around the TI TMS-8080, the Intel 8080A, and the Motorola 6800 machines. Cramer plans future products in this line for the RCA COSMAC, MOSTEK F8 and AMD 9080 chips. In each case, the kits come with complete engineering support documentation, information which is likely to be of interest to the serious microcomputer enthusiast. In addition to the documentation, the kits are said to contain the following items:

– The microprocessor IC itself.

– 1k by 8 bits of RAM.

– 1k by 8 bits of ROM in UV-erasable form, with a pre-programmed system monitor which provides a software front panel function.

– Four input and four output ports with complete parts requirements.

– Support circuitry including clocks, buffering, control, memory decode, etc.

– Controls and displays with enough parts for a hexadecimal LED front panel arrangement.

– Audio tape interface and a cassette filled with "useful programs."

It looks like a real product for the person who builds from the ground up. It is not a computer kit in the conventional sense of a MITS, Godbout or Southwest Tech product, but then it doesn't constrain you to a fixed wiring layout and system design either. You'll almost certainly need a wirewrap gun and lots of #30 wire to put this one together – or you could buy the PC board sets furnished by hobby suppliers like Southwest Technical Products. The $495 price is essentially for the semiconductors and documentation alone – the price does not include power supply, case or any packaging.

...CARL

The Newest Profession

"Computer hardware WIZARD is being sought for a new computer research lab at the University of Rochester." Further professional qualifications – must speak 6th century Arabic dialects, be able to compute *hex*adecimally, be able to zap gates, exorcise memory, burn ROMs with saltpeter and brimstone, and other related minor skills.

Submitted by A. M. Biguity

ALL WE CAN TELL YOU IS THAT MEN WHO DON'T SMOKE LIVE ABOUT 6 YEARS LONGER THAN MEN WHO DO SMOKE.*

If you want someone to help you
stop smoking cigarettes,
contact your American Cancer Society.

AMERICAN CANCER SOCIETY

** This fact taken from a research study is based on the smoker who at age 25 smokes about a pack and a half of cigarettes a day.*

8800 HARDWARE!

We have Altair compatible plug-in peripherals!

BUILD A SMART TERMINAL INTO YOUR ALTAIR!

Your Altair already has the intelligence, we provide the display module. This module is not a limitied "TV Typewriter" but an ultra-high speed computer terminal built into your computer. The VDM-1 generates sixteen 64 character lines from data stored in the 1K byte on-card memory. Alphnumeric data is shown in a 7x9 dot matrix format with a full 128 upper and lower case ASCII character set. The VDM-1 features EIA video output for any standard video monitor, multiple programmable cursors, automatic text scrolling and powerful test editing software included FREE! Available now.

!! MASS STORAGE !!

We have always wanted a low cost, reliable, fast access storage device using standard Phillips cassettes (we bet you have too), so we got to work and designed one – here it is! With the CDS-VIII Cassette Data System you have computer controlled access to 128K bytes of data within 20 seconds when using C-30 cassettes. We provide read/write electronics and transport controller, Altair interface, a case and power supply, and one or two multiple motor cassette transports plus FREE driving software! Yes, up to two cassette drives! Two drives provide much more powerful file handling and copying capabilities as well as, of course, twice the storage capacity. Data can be written and/or read asynchronously at any transfer rate up to 150 bytes/sec: at this rate 8K BASIC can be loaded in about 50 seconds! We have also included provision for use of any read/write electronic plug-in section so that tapes using HIT, Computer Hobbyist or Digital Group formats may be read at lower data rates. Available in November 1975.

4KRA Static Read/Write Memory

This 4096 word STATIC memory provides faster, more reliable and less expensive operation than any currently available dynamic memory system. The 4KRA permits Altair 8800 operation at absolute top speed continuously. All RAM's (Random Access Memories) used in the 4KRA are 91L02A's by Advanced Micro Devices, the best commercial memory IC on the market today. 91L02A's require typically 1/3 the power of standard 2102 or 8101 type RAM's and each one is manufactured to military specification MIL STD-883 for extremely high reliability. These memories can be operated from a battery backup supply in case of power failure with very low standby power consumption. (Ask for our technical bulletin TB-101 on power down operation.) In short we have done everything we could to make the best 4K memory module in the computer field, and because we buy in large quantity, we can make it for a very reasonable price. Available now.

2KRO Erasable Reprogrammable Read Only Memory Module

With this module the Altair 8800 can use 1702A or 5203 type Erasable Reprogrammable ROM's. The 2KRO accepts up to eight of these IC's for a capacity of 2048 eight bit words. Once programmed this module will hold its data indefinitely whether or not power is on. This feature is extremely useful when developing software. All necessary bus interfacing logic and regulated supplies are provided but NOT the EPROM IC's. Both 1702A and 5203 PROM's are available from other advertisers in this magazine for well under $25. Available now.

3P+S Input/Output Module

Just one 3P+S card will fulfill the Input/Output needs of most 8800 users. There are two 8-bit parallel input and output ports with full handshaking logic. There is also a serial I/O using a UART with both teletype current loop and EIA RS-232 standard interfaces provided. The serial data rate can be set under software control between 35 and 9600 Baud. You can use your old model 19 TTY! This module gives you all the electronics you need to interface most peripheral devices with the Altair 8800, it's really the most useful and versatile I/O we've seen for any computer. Available now.

MB-1 Mother Board

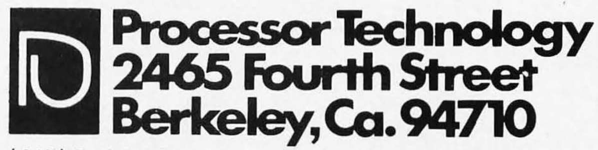
Don't worry any more about wiring hundreds of wires in your Altair to expand the mainframe. Our single piece 1/8-inch thick, rugged mother board can be installed as one single replacement for either three or four 88EC Expander cards, so you don't have to replace your already installed 88EC card if you don't want to. The MB-1 has very heavy power and ground busses and comes with a piece of flat ribbon cable for connection to the front panel board of the 8800, a built-in bus terminator, and card guide cage for sixteen plug-in slots. Available now.

PRICE LIST effective Oct. 1, 1975

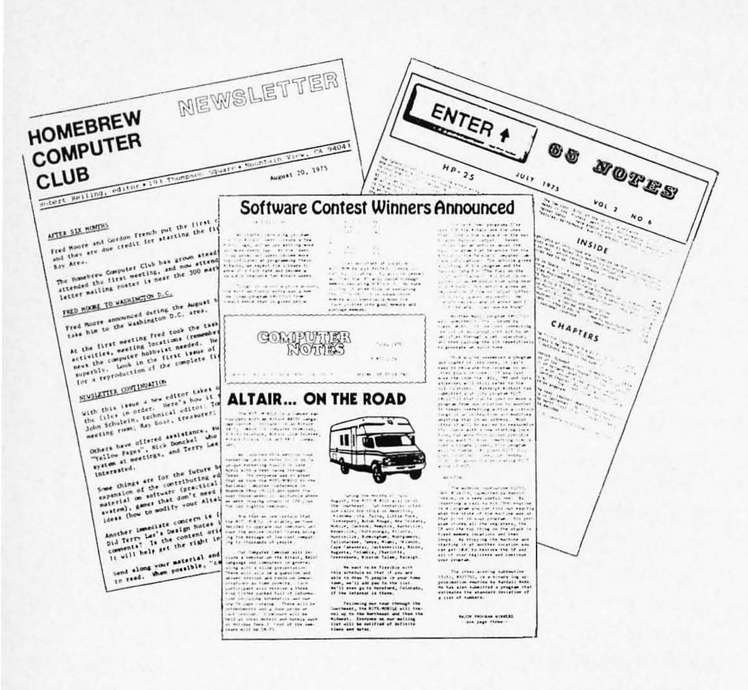
Item	Kit	Assembled	Delivery
2KRO EPROM module	$ 50.	$ 75.	3 weeks max.
3P+S I/O module	125.	165.	3 weeks max.
4KRA-2 RAM module w/ 2048 8-bit words	WRITE FOR DETAILS		
4KRA-4 RAM module w/ 4096 8-bit words	WRITE FOR DETAILS		
RAM only, AMD 91L02A 500 nsec, LOW POWER	8/$40.	–	3 weeks max.
CDS-VIII-1 Cassette Data System w/one transport	WRITE FOR DETAILS		
CDS-VIII-2 w/two transports	WRITE FOR DETAILS		
MB-1 Mother board, bus terminator, card cage	70.	–	3 weeks max.
VDM-1 Video Display module	160.	225.	3 weeks max.

TERMS: All items postpaid if full payment accompanies order. COD orders must include 25% deposit. MasterCharge gladly accepted, but please send us an order with your signature on it.

DISCOUNTS: Orders over $375 may subtract 5%; orders over $600 may subtract 10%.



(415) 549-0857



Clubs and Newsletters

The Evolution of the Homebrew Computer Club

BYTE has received the latest issue of the Homebrew Computer Club Newsletter, now edited by Robert Reiling, 193 Thompson Sq., Mountain View CA 94043. This club was started with a meeting on March 5, 1975, assembled by founders Fred Moore and Gordon French. Fred has been pretty much running the show since then via the newsletter activity. However, Fred's personal plans took him to the Washington DC area in the middle of August so he had to pass the ball along to someone else in the Silicon Valley. According to the newsletter dated Aug. 20, 1975, the following persons are now working on the club and newsletter:

- Robert Reiling, editor.
- John Schulein, technical editor.
- Tom Pittman, mailing list.
- Lenny Shuster, meeting room.
- Ray Boaz, treasurer.

The Aug. 20 issue is nicely done in a very professional photo offset format. It includes notes by John Schulein and Tom Pittman on the problem of audio cassette recording standards. (See BYTE's conference announcement elsewhere in this issue for some additional inputs.) The issue also includes comments by Ken McGinnis on TV display timing for the computer hobbyist, and some random data of interest compiled by Robert Reiling. The Homebrew Computer Club meets every two weeks at the Stanford Linear Accelerator Center, usually in the auditorium. Ask a guard for directions when you get there. Extrapolating the every two week algorithm from known dates of Aug. 20 and Sept. 3 gives the following dates in October and November: Oct. 1, 15, 29; Nov. 12, 26. The address for correspondence is temporarily Robert Reiling's site, but arrangements are being made for a club post office box.

HP-65 Users Club

The HP-65 is probably the one "home computer" in widest circulation by now. This is certainly true if your definition of "home computer" is something which costs under $1000, has programmability, uses microcomputer technology, has off-line storage, and some form of interactive set of user controls and displays.

The HP-65 Users Club is an organization founded by Richard J. Nelson (2541 W. Camden Pl., Santa Ana CA 92704) and devoted to the use of the HP-65, and other programmable calculators. The club publishes a newsletter, *65 Notes*, for a nominal fee. The July 1975 issue (Volume 2 number 6) is typical of the many issues included in a sample packet forwarded by Richard: a feature on the new HP-25, comments on the formation of various local club chapters, several programs, an article on repairs and/or modifications of the HP-65, and an article on how one reader of *65 Notes* adapted his HP-65 to a hard copy printer. (Note: *65 Notes* and the HP-25 Users Club are in no way affiliated with Hewlett Packard.) Here is where to turn for a wealth of information on programmable hand-held calculators.

Beta Iota Tau

Richard A. Petke announces formation of Beta Iota Tau, a new fraternity for campus computer freaks, knurds, hackers and assorted hangers on. For a full description see the letters section of this issue. His address:

BETA IOTA TAU
c/o Richard A. Petke
R.H.I.T. Box 520
Terre Haute IN 47803

If you want a definition of the terms "knurd," "hacker" and "computer freak," either look into a mirror or get in touch with Mr. Petke!

North Texas No-Name Club

The second informal gathering of the North Texas No-name Computer Club took place at the Irving Library on August 18.

Bill Fuller gave a short introduction on what the club's general aims could be and a few reasons for forming a group. With this, each of the 16 attendees introduced themselves, and summarized their computer interests and general background.

Lannie Walker, the Ft. Worth co-founder, brought his Martin MK-2 to display. Greg Walker demonstrated operation of his TVT-II. Ric Martin provided his TVT-I for comparison. Also available for a "look-see" were a Processor Technology In/Out board, ECS modem, two Suding cassette interfaces, ECS digital display PC board, and the MITS VLCT PC boards.

With introductions complete, a general open forum discussion took place until we were kicked out of the library at 9 pm. The "meeting" continued at a local coffee shop until 11:30 pm.

A quick summary of pertinent data obtained from questionnaires filled out shows:

Computer: Five Altairs, one MK-2 and one home brew.
TVT: One TVT-I and two TVT-II.
Cassette interface: Three Suding, one ECS, one Computer Hobbyist, one MITS.
Paper Tape: Three punches, one reader.
Teletype: Five of various vintage.

Since we are still somewhat unstructured as a formal club, interested "byters" should contact Bill Fuller, 2377 Dalworth 157, Grand Prairie TX 75050, 1-214-264-0111 or Lannie Walker, Route 1, Box 272, Aledo TX 76008, 1-817-244-1013.


WORLD'S FIRST COMPUTER STORE

COMPUTER STORE

COMPUTER NOTES

MITS-MOBILE HITS SE

Altair User's Group

In the words of *The Agency,* MITS' advertising subsidiary, "The Altair User's Group is quite possibly the largest hobbyist organization

Information Parlor). The store sells Altairs over the counter of course, and functions as a general computer hobbyist gathering place and information center. *Computer Notes* also features a travelogue of the MITS-MOBILE Altair Caravan, which toured the Southeast during August and September.

In his editorial, Dave Bunnell leads a good deal of rumors to their proper resting places, covering "off brand peripherals, memory cards, etc.", the false rumors concerning "less than full spec" Intel 8080 chips, software agreement technicalities, MITS' development of a Motorola 6800 system, and delivery complaints. Dave precedes all this with the explanation, "One point that has gotten us good press in a number of publications is that we try

future plans; Monte Davidoff, one of the authors of Altair BASIC, illustrates the string handling and recursive subroutine capabilities of BASIC in "Fun with Altair BASIC."

Monte does a good job of describing these interesting possibilities of BASIC, even though such applications can be a bit strenuous for both the reader and the BASIC language itself, as witnessed by Monte's closing comment on factorial computation: "If confusion still prevails, do not worry about it."

Bill Gates, in his article "Software Hints for 8800," gives just that. Mentioning some of the reasons for the 8080's power, Bill goes on to discuss binary coded decimal (BCD) arithmetic, giving a sample routine for conversion from BCD to binary. After a short lesson on special short branching ("skip") techniques

like an ad for BYTE, Dave is pleasantly positive in his remarks.

I have a couple of complaints. The format of *Computer Notes* is imaginatively done, but at some points the text is hard to follow – you don't know what to read next. [But then, it is only fair to point out that BYTE may suffer the same malaise in one or two places . . . Carl] Also, a lot is assumed on the part of the reader. I realize that any such newsletter can't function as a tutorial publication (that is a major part of BYTE's job) but the "HARDWARE" and "SOFTWARE" sections would certainly throw any real beginner. But I judge too harshly: *Computer Notes* is truly a good bulletin for the users of Altair equipment.

To paraphrase Dave's closing question in his BYTE review, is the Altair User's

BYTE #2, p. 36 LIFE Line 2, Fig. 3. The variable DONE is incorrectly described.

Correct description:

DONE – the variable set by KEYBOARD_INTERPRETER after a user command to end execution.

BYTE #2, p. 38, LIFE Line 2, Fig. 6. The variables NCMAX, NCMIN, NRMAX, and NRMIN are incorrectly classified as local – used only by GENERATION. They should have been classified as shared with the whole program, i.e., global.

BYTE #1, p. 54, Fig. 5, Write Your Own Assembler. Dan Fylstra caught a bug in his 8080 example about the day after BYTE #1 went to press – and several readers (including Bill Gates of MITS Altair BASIC fame – see letters) caught it. The mistake is that branch table has 3 bytes per entry and the example multiplies by *2* with a left shift (RLC). The corrected code would be:

"NEXT TOKEN" ROUTINE

```
LXI    H,BRTAB  H – branch table base

LDAX   D        get translated char from line

RLC             times 2 for branch table index ? almost
MOV    B,A      save "2n" in B
LDAX   D        fetch "n" again . . . translated character
ADD    B        "3n" = "2n + n" . . .

   .
   .
   .
rest of NEXT TOKEN
   .
   .
   .
```

One final word – this is an example of a "patch". As programs grow and patches multiply the result can get hairy. There is no pretension that this is the optimal code for the next token algorithm; one of the challenges of programming is to see how close one can actually come to optimal code.

BYTE #1, p. 16. Author James Hogenson supplies the following correction to information on the RGS-008A computer:

The two LEDs not used by the upper memory address are *not* used to indicate the second and third bytes of a two or three byte instruction. They indicate the *control code* which varies in a two or three byte instruction. One machine cycle is required for each byte of an instruction. The first byte of an instruction is always an instruction fetch cycle indicated by control code 0. The second and third cycles are for data writing, reading or for an I/O operation. Control code 1 indicates an I/O operation, control code 2 (octal) indicates a memory data read cycle, and control code 3 indicates a memory data write cycle.

The power supply produces *positive* 5 V at 5 Amps, and –12 at 1 Amp.



Don't Forget the Pinouts of These Memory Circuits – They're All Very Useful Both in Building More Memory for Kit Products and as Memory Elements of Your Own Designs . . .

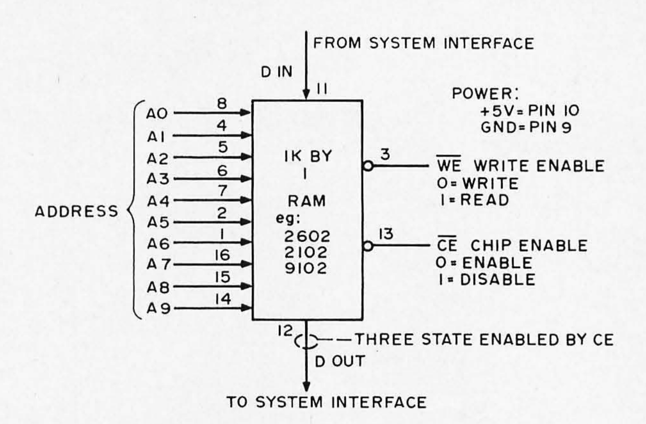


Fig. 1. The 2102, alias 9102, alias 2602. This memory circuit is the most generally useful item on the market. You will find it in graded price ranges depending upon speed; for the 8008 and other slow processors, the low speed and inexpensive versions with access times up to 1 microsecond can be used. For the later generation microcomputers such as the 8080, M6800 or MCS6501 or PACE, higher speed versions are recommended in most cases if you want to take advantage of full processor capability. For complete specs, see the manufacturer's specs. Here is a pin diagram of this memory design. To use it, present addresses then Chip Enable should go to logic 0 (ground level), then after the access time delay you can either read the output data lines, or write what you've presented on the input data line. The write operation is performed by a low-going pulse on the WE line, with a minimum duration of from 300 to 750 nanoseconds depending upon the speed rating of the chips you buy.

**NOTE: When talking about logic and memory, the notation "K" means 2^{10} or 1024, not 10^3 or 1000.*

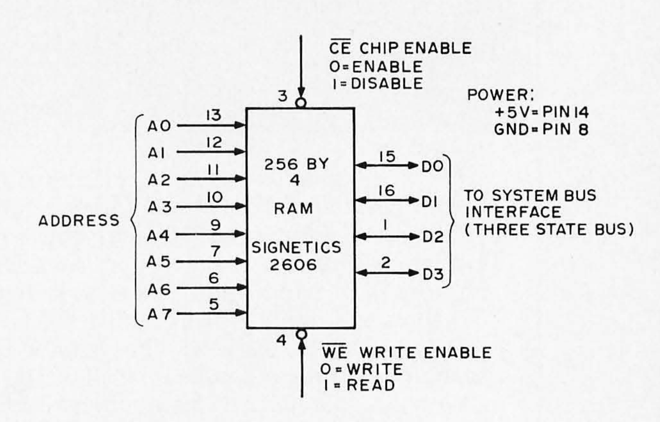


Fig. 2. The Signetics 2606 part is a variation on the static memory concept which uses a parallel 4-bit word orientation. There is a total capacity of 1024 bits in 256 words. For reading out of this memory, you hold the Write Enable (WE) line at logical 1, select an appropriate combination of address inputs, then drop the Chip Enable (CE) to logical 0 to enable the output bus drivers. To write into the memory, you enable the chip as if to read, but drive the data bus from a source such as your CPU while dropping the Write Enable line to the logic 0 (ground) level for a short period (minimum 400 ns). The spec sheet for the 2606 gives an access and cycle time of 750 ns, which means that it is on the hairy edge of working with one of the 1 microsecond cycle second generation microprocessor CPUs. Here is the pinout of the 2606 part.

Shipmates wanted.

to join 'barefoot' expedition to West Indies Islands.



And, we'll promise you only one thing. The most adventurous vacation you have ever experienced. Hop aboard our schooner for 10 sunfilled days and moonlit nights. A congenial group of shipmates now forming and setting their heading for Saba, Grenada, St. Lucia, Guadaloupe. And, Dutch St. Maarten. And, French Martinique. Explore pink, white and black sand beaches. Scale the cliffs, climb a volcano, prowl old forts and quaint towns. Hoist the sails, take a turn at the wheel or feet on the rail and let us do the work. Then limbo to a steel band and fall asleep under a star spangled Caribbean sky. Share from $250. If you have the spirit... come Windjammin'.. a true life adventure. This coupon will bring you a full color adventure brochure.

Send me your color brochure on 10 day 'barefoot' vacations from $250.

Name ____________________

Address ____________________

City ____________ State ____________ Zip ________



Post Office Box 120, Dept. 00, Miami Beach, Florida 33139.

What Are We Getting Into?

While many hobbyists tend to think in terms of their own uses for microprocessors (uP) such as games, music, art, index systems, mailing lists and such, we shouldn't forget the enormous business applications for these systems once they are commonly available.

We can see uP systems with 1500 megabyte storage (or larger) which would make very economical packages for lawyers, for example. Entire law libraries could be right at hand with access taking less than a minute to any case. Commonly used legal forms could be filled out via video terminals (VT) and then printed on a line printer. Complete records of all clients would be immediately available, plus billing, phone calls, etc.

How about a uP system for doctors? The preliminary diagnostic work could be done by a receptionist during the heel cooling period in the waiting room, and this might even include temperature, heart, blood pressure, and a few other sensors hooked into the uP. The VT would be used with a diagnostic program to ask the questions. No disease, no matter how rare, could evade a really good uP diagnostic system. And the uP would print out the prescription and instructions for the patient.

You can work up your own dream of inventory controls for shoe stores, drug stores, and other businesses.

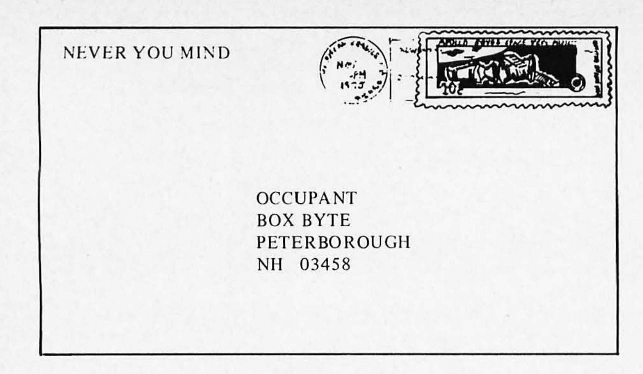
Considering the applications on the horizon, one can understand why so many firms are getting that glint in their eye when they think about making hardware for it. It seems likely that there will be more uP systems sold during 1976 than there are computer systems in the country presently in use.

While we are thinking in terms of $500 uPs at present, this may go down to $250 and then even $125 for the processors. VTs are presently running about $350 and could get to half that in time. Floppy disk systems are being announced in the $2000 range, hinted at in the $1000 range, and predicted to drop to $500. Bulk memory systems using some new tape cartridges promise to provide enormous data bases with access times of 20 seconds maximum and an average retrieval time of 10 seconds. This might not satisfy Avis or American Airlines, but it is superb for a legal assistant researching cases.

If you are plugged into any developments of interest to BYTE readers please consider this an open invitation to pass along the information as soon as possible.

BYTE hopes to provide a stabilizing influence, encouraging standards, compatability, and harmony in the uP field.

—Wayne Green



A BIT OF NYBBLING AT BYTES IN A WORD

Dear Sir:

I was very delighted to receive the first issue of BYTE. I found all the articles, and even the letters interesting, absorbing, and amusing. There are, though, three points about which I would appreciate some information and clarification.

First of all, I seem to be a bit confused about your use of the words "word", "byte" and "bit". As I understand their usage, "word" refers to the grouping of "bytes" which are considered to be one memory location, and "byte" is a grouping of bits, three or four depending on whether octal or hexadecimal is used. The article "Which Microprocessor for You?" seems to use the terms ''word'' and ''byte'' interchangeably, which confuses me about the processor's location referencing ability. Who is right, me or you?

Secondly, I have seen information about 8-bit and 16-bit microprocessors, but no ads about 12-bit microprocessors. Having gotten my programming training on a 12-bit system (PDP-8/L) may have made me unduly partial to 12-bit words, but an 8-bit system would not allow me to cheat and store two ASCII characters in a location, while 16 bits seems too big for my desires. Does anybody put out a 12-bit microprocessor with 4k memory?

Finally, I see a large number of keyboards out on the market; I do not see any printers, line or teletype type. Where are they hiding?

Thank you for your attention.

Stephen Holland
Glenview IL

Thank you for your letter, Stephen. Precision in description of concepts is one of the most powerful techniques in the human race's evolutionary bag of tricks. The only hitch is communicating what the definitions are when there are many different people involved and each has a unique past experience. Hopefully in written communications we can keep the ambiguities (other than occasional fun) to a minimum. To clarify your very valid complaint about memory terms, the following are some fairly fundamental definitions. When you read certain articles, such as Hal Chamberlin's, it turns out that two of these definitions are equivalent. For other situations they may not be:

**A bit is the fundamental – it is the unit of storage. All other terms are described in terms of bits. One memory cell can hold one bit's value of logical zero or one.*

**A byte is a grouping of bits in the organization of memory, usually defined ala IBM as 8 bits. (Nybbles are half-bytes.)*

**A word is the generic and very general term for "n" bits at once in a computer's memory. The term word is the same as byte when referencing an 8-bit byte-oriented machine, but will certainly differ for a 12-bit PDP-8 or a 16-bit PDP-11 or some old 36-bit Univac clunker or a 60-bit CDC machine.*

If that's not enough to confuse the issue, some of the early literature on machines such as IBM's old STRETCH computer referenced variable sized bytes extending the definition to "any group of n bits".

In general, though, a given machine will have a "memory address space" for direct reference to a particular size of ''word''. For the byte-bangers (8080, 6800, 8008, 6501, F8, etc.) each memory address gets you one byte. For a PDP-8 each address in memory gets you one 12-bit word. And so on.

As far as 12-bit micros go, there are none which are generally available to amateurs yet. There is the CMOS PDP-8 replacement which Intersil makes, but it is very expensive as yet. There is the FABRI-TEK processor which is an MSI copy of a PDP-8 at about $1000 for the CPU, but again a working system is not inexpensive. The FABRI-TEK machine does come with 4k built in at the price.

Concerning printers, you are again getting up in price. A simple receive only Teletype is about $200-300 if you get to the right place at the right time in the surplus market. There is something called a Creed Teletype which I have seen some literature on – an old 5-bit Baudot machine which you could probably hook up fairly easily to Bill Godbout's PACE processor with its ASCII-Baudot software. Where are the printers hiding? In surplus houses which don't advertise 'em because their stock changes too frequently.

. . . Carl

Dear Carl,

I just finished reading the first issue of BYTE, received today. I found the articles interesting but now realize how little I know about programming terms and functions.

I've been in ham radio and radio and television broadcast engineering for well over 15 years. I understand the basics of digital electronics and have designed control circuits and enjoy using digital techniques in other types of electronic circuitry.

I can follow the articles until the author starts hitting me with programming terms, then I'm lost. Perhaps you might try to educate such as I with some very basic articles on programming and those hardware functions mostly limited to computers. You might also recommend some up-to-date books on these subjects.

I would very much like to learn more about this field. Perhaps I'm not the typical reader you are publishing BYTE for, but I suspect there are other readers out there with the same problems.

I'd also like to see you insert a bingo card to make it easier to get further information and catalogs from your advertisers.

Kenneth Knecht
Chicago IL

Thank you for your input, Ken. I am indeed worried about the problem of acquainting the novice readers of BYTE with the terminology and concepts of home brew computing. I'm working on getting various authors to submit tutorials on the concepts of programming for people such as yourself with a strong hardware background. Then there is the analogous problem from the other side – explaining the mysteries of hardware to software people. For instance, a software-oriented friend of mine

pointed out that he got lost on the diode matrix concept in the keyboard article of BYTE #1, a concept which is probably familiar to a number of readers with a hardware background. As long as I am enumerating possible approaches to the art of home brew computing, I shouldn't forget the person who has neither hardware or software background — the complete novice.

The way I can find out about what "the typical reader" is thinking is for "the typical reader" to send in comments, as you have. One of the biggest unknowns I had when I first came up with the problem of editing this magazine was figuring out in my mind just what the readership is: Is it a bunch of hackers from the MIT or SRI artificial intelligence labs? Hardly — although I know a few who subscribe. Is it the group of ambitious people who started 8008 projects when they first came out? Yes — but not exclusively, by a long shot. Is it the engineer who always wanted his own hardware — but never heard of software engineering? Yes — some readers fall in that realm. Is it the applications programmer who gets kicks out of programming and always wanted a processor of his own? Of course — there is more than one COBOL programmer in the audience to say nothing of FORTRAN people and PL/1 people.

Is it the BASIC fundamentalist who is lost in his applications games and suddenly discovers there is more to computing than GOSUB and LET? Sure — and he has a whole realm of hardware and software to learn about. The problem is a tough one — but constructive input from readers in the form of articles and opinion is an important key to shaping the content of the magazine. Introductory materials and articles to get people acquainted with terminology will be in future issues — as you'll find to be true in this issue as well.

. . . Carl

WHO IS STEPHEN B. GRAY?

Dear Sir,

Your description of the Amateur Computer Society (ACS) in the Clubs column of the first issue of BYTE is strangely constructed. Please allow me to supply your readers with further info about the ACS along with some history of the home computer "hobby."

Stephen Gray organized the ACS in 1966 to serve everyone who is involved in designing and building his own home computer. The ACS is an informal organization, with no such thing as officers or meetings, since its members are widely separated across the U.S. The main function of ACS has been to initiate the exchange of ideas and equipment among its members. This is done by the *ACS Newsletter*, which describes the plans and accomplishments and acquisitions of the various members. It also lists the sources of surplus computer parts available for the hobbyist. Back in the late 60's and early 70's there were no kits available to buy, and not too much surplus computer equipment. This meant that most home computer builders had to design their CPUs and I/O interfaces from scratch. The ACS in those years was made up mostly of digital circuit and systems design engineers. It was like the early decades of amateur radio, when not much commercial equipment was available for hams to buy.

Today there are several hundred ACS members representing all kinds of careers in the electronics and computer industries. Lots of them have had operating home computers for five to eight years! These people have a really wide variety of home computers, from old vacuum tube ones and all discrete germanium transistor ones to microprogrammable LSI minis and microprocessor based ones. Some of them have reconditioned obsolete models that you can't even buy parts for anymore, with such weird peripherals as drum memories and flexowriters. Others have lots of the latest stuff, like CRT graphics terminals, digital cassettes, and floppy disks. And of course lots of them have designed their own CPU architecture and instruction sets to provide special purpose capabilities that are unmatched by anything in the industry.

Then any article published for one of these systems will be quite useful to thousands of people.

But as it is now, there is little in the line of construction articles and detailed program descriptions (with source code listings) that could be directly used by more than a handful of ACS members. Therefore, Stephen Gray does not publish such articles in the ACS newsletter. Instead he has used this disparity as an opportunity to make it a more personal newsletter, reporting in each issue what various individuals have accomplished, what difficulties or experiences we have written about, who of us have information or equipment to sell, or trade, who needs information about certain equipment, what new items are available commercially, to the home computer market and so on. All in all it's a very helpful and informative newsletter.

"The ACS is an informal organization, with no such thing as officers or meetings . . . the main function has been to initiate the exchange of ideas and equipment . . ."

With such a wide variety of approaches, design philosophies and languages existing in the home computer field, each ACS member's computer "lives" in its own special world, having little in common with the next member's computer (which may be several dozen miles away). Compare that with tomorrow's home computer scene, where there will be thousands of people with systems built around audio cassette mass storage, VDTs and 6800s and 8080s.

I feel that Mr. Gray has done an excellent job of bringing us home computer builders together, and I believe that the ACS and its members have quite a lot to offer to the new generation of home computer enthusiasts that have been spawned by the age of the microprocessor. I think that the recent developments in our field are just great and the near future promises to be absolutely fantastic! It all can't come soon enough for

me! Even though science fiction writers predicted it long ago, who knows what the future of computing will bring? I certainly welcome BYTE into our field and wish you the best in this new computer age.

Dick Snyder
Chelmsford MA

Strangely constructed? Indeed! The goal of obtaining more information was accomplished, Dick, although Mr. Gray is too modest to send it to BYTE himself. All I had to go on was one cryptic letter Stephen B. Gray had written to Wayne Green and a file reference supplied to me by Richard Gardner who makes it his business to know what's going on where and by whom.

I am firmly of the belief that anyone who has made verifiable contributions to the technological progress of this extremely interesting hobby should be recognized – as you point out in your letter Mr. Gray has made such a contribution by serving as a focal point for a group of the pioneer home computer hackers. To the best of my knowledge he is the earliest person to put together a vehicle for communications amongst home brew computer people, a notable contribution.

. . . Carl

The comparison of simple subroutines for different microcomputers does not show the relative merits of the machines.

WHICH MICROPROCESSOR EVALUATOR FOR YOU?

Dear Editor,

I am pleased to see a high quality magazine for the computer hobbyist. In this large and fast changing field BYTE will be invaluable as an educational and informational forum.

I have spent the last three years programming microcomputers, most recently writing ALTAIR BASIC with Paul Allen and Monte Davidoff, and hope to share some of what I've learned with your readers.

I do have some disagreement with statements made in Hal Chamberlin's article "Which Microprocessor for You?" Of the three micros, the IMP-16, the Intel 8080 and the Intel 8008, the 8080 has the best memory efficiency due to the number of things that can be done in a single 8-bit instruction. The IMP-16 is second best, requiring 30% more memory bits; and the 8008 is a distant third, requiring about twice as much memory as the 8080.

The IMP-16 is the fastest machine with the 8080 a close second. The 8008 is at least 12 times as slow as the 8080. This is because the 8008 not only has a much slower cycle time, but it requires many more operations to perform an equivalent function.

Software from MITS costs about six times as much if hardware is not purchased from MITS. As long as hardware is purchased from MITS, MITS is willing to just break even on the software.

The comparison of simple subroutines for different microcomputers does not show the relative merits of the machines. The best way someone who doesn't know a lot about software can judge the different microcomputers is by seeing how large and how fast a large program like a BASIC interpreter or FORTRAN compiler written by a professional programmer will run on the machine.

William H. Gates
President, Micro-Soft
Albuquerque NM

PS: The program on page 54 of BYTE #1 doesn't work since the dispatch table entries are three bytes long instead of two.

The substance of Bill Gates' letter is that the problem of evaluating a computer's performance is a complicated issue. Every machine has its little idiot-syncrasies which enable it to outperform its competitor in a specific context. The tests one uses to evaluate the machine in many ways affect one's results: Bill (like myself) would tend to evaluate a machine based upon a complicated "benchmark" application or systems program. This sort of evaluation typically will exercise the full range of the CPU's instruction set in a real-world large program context. Other people would tend to evaluate not upon that benchmark, but on the ease of generation of machine code which is fully optimized by a modern compiler. The compiler writer's benchmark is different from the systems hacker's benchmark. Then there is the small computer world – which is the best computer from the home brew hacker's standpoint? Probably in the current state of the art it is the machine which has the easiest instruction set to deal with given a modicum of support in the form of assemblers from the kit supplier.

People have been evaluating computer instruction sets since the first time some disgruntled programmer cursed at an IBM 650 (or earlier) product. And people will continue to do so. I'd like to see some of BYTE's readers giving personal evaluations of their own experiences programming some of the machines which are now available. This will produce still another form of benchmark.

Oh yes, regarding the mistake on page 54 – see the "Diagnostics" heading in the Nucleus of the Queue, in this issue.

. . . Carl

IT'S GREEK TO ME . . .

Carl,

With all due respect, your dump of computer organizations in the last issue of BYTE was incomplete! (Obviously a hardware bug.) You omitted perhaps the biggest one of them all (at least in potential) – Beta Iota Tau (BIT). Yes fans – that's right – it's here now – the first computer science college fraternity. Us knurds got our (expletive deleted) together this time. You say you never heard of us? That's understandable, we're new (and not fully established yet). As of the present only two chapters are emerging from the mass of red tape (7, 9 and Cassette): the founding chapter here at Rose-Hulman Institute of Technology (and software disproving grounds), and the Beta chapter at Michigan State. As the probability of this publication finding its way to college campuses this fall approaches one as a limit, we would like to take this time to rip off some advertising space.

Are you interested in starting a chapter of BIT at your college or university? If you are, then ASCII some of your friends. Show them this crazy letter we're writing at this crazy hour. Drop us a

line (RS-232 compatible) at the address below. We'll send you more information as we get it ready. If nothing else we'll help synchronize you with the rest of the knurds at your school (provided they write in also). Write to:

000 * BETA IOTA TAU
001 * c/o Richard A. Petke
002 * R.H.I.T. Box 520
003 * Terre Haute IN 47803
For BIT,

R. Petke
High Order Bit
B. Copus
Chairman of the Vice
L. Passo
I/O Processor

I suppose it just had to happen some time. I'm glad to hear about BIT, Richard. The very idea of BIT leads to all manner of intricate and exciting possibilities. Why, to start off with, you'll need to recruit a bunch of novices known as "Least Significant Bits" into which they have been indoctrinated with their Initialization Writes, they can work their way up through the ranks. But you will have to be selective about it — you'll have to watch out for shifty characters. AND in the fall you can schedule (at high priority) some major social events in the BIT house culminating in the Masked Ball around Halloween. OR in the spring you can have another big social affair at which you'll crown the Queen of the PROM (hopefully she'll not get burned). Speaking of such matters, to be fair you had better organize the logical complement of your organization. Find some ladies to form a sorority, or change your parameters and get some grand scale integration into your organization. Keep me posted on progress of BIT.

... Carl

from page 7

be unused. In general, as many of these locations as you can afford should be filled up with random access memory chips, which, experience has shown, people are always able to use up in programs. Sooner or later you will find yourself limited by the constraints of small memory! For the benchmark system, the minimum random access memory should be 4k (4096) 8-bit bytes or 2k 16-bit words. A preferable number is 8k bytes or 4k 16-bit words.

ROM Systems Software?

How do I get my first programs into memory after turning on power? The answer to this question is the method of "bootstrapping" or "initial program loading" (IPL) which is used by a computer. Early in the minicomputer game, technology of computing was at a state where the principal bootstrapping method was a set of front panel switches which addressed memory locations and allowed the programmer to put in short programs by hand.

With the advent of the new high density ROM integrated circuits, it is now possible to provide the convenience of an automatically bootstrapped system through systems software which is cast into the concrete form of an ROM device.

Many of the kit suppliers I have talked to are either currently supplying or intending to add this ROM systems software feature. Initially, the programs which are "built-in" tend to be fairly standard "control panel" type routines which use a terminal (Teletype or television typewriter) for a set of simple commands. Later – with inputs from users regarding desirability – you can expect to find prepackaged assemblers and high level language compilers/interpreters occupying major portions of the address space available in typical microcomputers. This will make the systems software feature even more versatile.

Keyboard and Displays?

But of course. The interactive nature of an editor capability cannot be realized with a mere control panel. The same thing goes for most of the more interesting applications of the small computer. You will need a character-oriented display device and a typewriter style input – whether these be a TV typewriter or an old Baudot coded Teletype clunker is up to you. The typical programs will be controlled by keyboard commands and will produce outputs back to the display.

Cassette Tape Interfaces – Mass Storage Without Mass Dollars

Mass storage is a definite must item for the small computer system. But traditional industry peripherals tend to be expensive, starting at the low end with digital cassette drives and floppy disks at about $500-$800, and working upwards. The solution is to adopt an audio recording method which uses inexpensive ($50) cassette recorders and appropriate interfaces. This allows you to perform the editing benchmark function while keeping the total system cost low. I'll have more to say on this subject later in this editorial. A minimum of two such tapes is required for a decent editor, because one must be set to "read" old data, and the second must be set to "write" new edited data resulting from your changes. Three is a more desirable number still if you want to do "sort/merge" applications, but two will suffice for the editing benchmark.

Suppose Your Budget is Limited – Can It be Done in Stages?

What I have just described is the minimum necessary equipment for a fully functional implementation of the small computer benchmark capability, editing. Modularity rules in the computer world, however, so you can easily start out with less function and work up to the benchmark capability in time. You'll also probably end up exceeding this benchmark of hardware/software capability after a while; modularity does not stop at this level of function. The basic place to start is with a CPU – it'll not be much more than a blinking light box without peripherals, but that's enough to show that "it works" Then, you can add on the interactive keyboard/display of some sort, along with memory (presumably the ROM software came with the CPU). Finally, you can add on the tape interfaces and additional memory in order to arrive at the full benchmark capability. From then on, you can enhance the system with new peripherals and more memory until you end up with a very capable system which can run full BASIC, a decent systems programming language compiler, and all the games, practical applications and amusements you can dream up for the computer.

Experience has shown that sooner or later you'll feel constrained by any size of memory – the greed of many programmers for more memory is unbounded!

BYTE's Ongoing Monitor Box (B.O.M.B.)

BYTE would like to know how readers evaluate the efforts of the authors whose blood, sweat, twisted typewriter keys, smoking ICs and esoteric software abstractions are reflected in these pages. BYTE will pay a $50 bonus to the author who receives the most points in this survey each month. The following rules apply:

1. *Articles you like most get 10 points, articles you like least get 0 (or negative) points – with intermediate values according to your personal scale of preferences.*
2. *Use the numbers 0 to 10 for your ratings, integers only.*
3. *Be honest. Can all the articles really be "0" or "10"? Try to give a preference scale with different values for each author.*
4. *No ballot box stuffing: only one entry per reader!*

Fill out your ratings, and return it as promptly as possible along with your reader service requests and survey answers. Do you like an author's approach to writing in BYTE? Let him know by giving him a crack at the bonus through your vote.

Page No.	Article	LIKED LEAST … BEST
12	Lancaster: Ins and Outs . . .	0 1 2 3 4 5 6 7 8 9 10
20	Wadsworth: Computers Are . . .	0 1 2 3 4 5 6 7 8 9 10
36	Wier: Hexpawn . . .	0 1 2 3 4 5 6 7 8 9 10
42	Gipe: Computers . . .	0 1 2 3 4 5 6 7 8 9 10
52	C. Helmers: Notes . . .	0 1 2 3 4 5 6 7 8 9 10
56	Fylstra: Son of Motorola	0 1 2 3 4 5 6 7 8 9 10
64	Nico: Monitor 8½ . . .	0 1 2 3 4 5 6 7 8 9 10
66	P. Helmers: Versatile ROM . . .	0 1 2 3 4 5 6 7 8 9 10

5% OFF ON ORDERS OVER $50.00
10% OFF ON ORDERS OVER $100.00
15% OFF ON ORDERS OVER $250.00

NOVEMBER SPECIALS

CALCULATOR CHIPS

5738 8 digit multiplexed — five function — chain operation 2 key memory — floating decimal — independent constant — interfaces with led with only digit driver — 9 V batt. oper. 24 pin **$3.95**

5739 9 digit — 4 function — chain operation — 9V battery operation — 16 sec turn-off 22 PIN DIP **$3.95**

CLOCK CHIP

5311 6 digit multiplexed — 50-60 Hz — BCD and 7 seg out — fast, slow set 12-24 Hr — 28 PIN DIP **$3.95**

TIMERS

555	Multipurpose timer 8 PIN DIP	$ .69
556A	Dual 555 14 DIP	$1.29

TTL (DIP)

7432	$ .19	7493	$ .49
7448	.89	74107	.33
7475	.45	74121	.35
7490	.49	9601	.75

POCKET CALCULATOR KIT

5 function plus constant — addressable memory with individual recall — 8 digit display plus overflow — battery saver — uses standard or rechargeable batteries — all necessary parts in ready to assemble form — instructions included



CALC KIT (WITH BATTERIES) **$12.95**
BATTERIES ONLY (DISPOSABLE) SET **$2.00**

9 DIGIT LED DISPLAY FNA 37



On multiplexed substrate, comm. cathode compatable with all 8 digit calculator chips, 7 segment right hand decimal, red with clear magnifying lens, .12" character, 1 to 4 MA, 1.8 V typ 2⅜" x ¼" x 3/16" high **$3.95**

VOLTAGE REGULATORS

309K	TO-3	$1.00
340T-5V	TO-220	$1.00
340T-15V	TO-220	$1.00

MEMORIES

1103	$ 1.29
5203	12.95
5260	.99
5262	3.19

TTL

		7451	.17	74154	1.25
7400	$ .14	7453	.17	74155	1.07
7401	.16	7454	.17	74156	1.07
7402	.15	7460	.17	74157	.99
7403	.16	7464	.35	74158	1.79
7404	.19	7465	.35	74160	1.39
7405	.19	7470	.30	74161	1.25
7406	.35	7472	.30	74162	1.49
7407	.35	7473	.35	74163	1.39
7408	.18	7474	.35	74164	1.59
7409	.19	7475	.57	74165	1.59
7410	.16	7476	.39	74166	1.49
7411	.25	7483	.79	74170	2.30
7413	.55	7485	1.10	74173	1.49
7416	.35	7486	.40	74174	1.62
7417	.35	7489	2.48	74175	1.39
7420	.16	7490	.59	74176	.89
7422	.26	7491	.97	74177	.84
7423	.29	7492	.71	74180	.90
7425	.27	7493	.60	74181	2.98
7426	.26	7494	.94	74182	.79
7427	.29	7495	.79	74184	2.29
7430	.20	7496	.79	74185	2.29
7432	.23	74100	1.30	74187	5.95
7437	.35	74105	.44	74190	1.35
7438	.35	74107	.40	74191	1.35
7440	.17	74121	.42	74192	1.25
7441	.98	74122	.45	74193	1.19
7442	.77	74123	.85	74194	1.25
7443	.87	74125	.54	74195	.89
7444	.87	74126	.63	74196	1.25
7445	.89	74141	1.04	74197	.89
7446	.93	74145	1.04	74198	1.79
7447	.89	74150	.97	74199	1.79
7448	1.04	74151	.79	74200	5.90
7450	.17	74153	.99		

LOW POWER TTL

74L00	$ .25	74L51	$ .29	75L90	$1.49
74L02	.25	74L55	.33	74L91	1.45
74L03	.25	74L71	.25	74L93	1.69
74L04	.25	74L72	.39	74L95	1.69
74L06	.25	74L73	.49	74L98	2.79
74L10	.25	74L74	.49	74L164	2.79
74L20	.33	74L78	.79	74L165	2.79
74L30	.33	74L85	1.25		
74L42	1.49	74L86	.69		

HIGH SPEED TTL

74H00	$ .25	74H21	$ .25	74H55	$ .25
74H01	.25	74H22	.25	74H60	.25
74H04	.25	74H30	.25	74H61	.25
74H08	.25	74H40	.25	74H62	.25
74H10	.25	74H50	.25	74H72	.39
74H11	.25	74H52	.25	74H74	.39
74H20	.25	74H53	.25	74H76	.49

8000 SERIES

8091	$ .53	8214	$1.49	8811	$ .59
8092	.53	8220	1.49	8812	.89
8095	1.25	8230	2.19	8822	2.19
8121	.80	8520	1.16	8830	2.19
8123	1.43	8551	1.39	8831	2.19
8130	1.97	8552	2.19	8836	.25
8200	2.33	8554	2.19	8880	1.19
8210	2.79	8810	.69	8263	5.79
				8267	2.59

9000 SERIES

9002	$ .35	9309	$ .79	9601	$ .89
9301	1.03	9312	.79	9602	.79

CMOS

		4016A	.56	4050A	.59
4000A	$ .26	4017A	1.19	4066A	.89
4001A	.25	4020A	1.49	4068A	.44
4002A	.25	4021A	1.39	4069A	.44
4006A	1.35	4022A	1.10	4071A	.26
4007A	.26	4023A	.25	4072A	.35
4008A	1.79	4024A	.89	4073A	.39
4009A	.57	4025A	.25	4075A	.39
4010A	.54	4027A	.59	4078A	.39
4011A	.29	4028A	.98	4081A	.26
4012A	.25	4030A	.44	4082A	.35
4013A	.45	4035A	1.27	4528A	1.60
4014A	1.49	4042A	1.47	4585A	2.10
4015A	1.49	4049A	.59		
74C00	$ .22	74C74	$1.04	74C162	$2.93
74C02	.26	74C76	1.34	74C163	2.66
74C04	.44	74C107	1.13	74C164	2.66
74C08	.68	74C151	2.61	74C173	2.61
74C10	.35	74C154	3.15	74C195	2.66
74C20	.35	74C157	1.76	80C95	1.35
74C42	1.61	74C160	2.48	80C97	1.13
74C73	1.04	74C161	2.93		

MEMORIES

1101	256 bit RAM MOS	$ 1.50
1103	1024 bit RAM MOS	3.95
1702A	2048 bit static PROM UV eras.	17.95
2102	1024 bit static RAM	4.25
5203	2048 bit UV eras PROM	17.95
5260	1024 bit RAM	2.49
5261	1024 bit RAM	2.69
5262	2048 bit RAM	5.95
7489	64 bit ROM TTL	2.48
8223	Programmable ROM	3.69
74200	256 bit RAM tri-state	5.90

CALCULATOR & CLOCK CHIPS

5001	12 DIG 4 funct fix dec	$2.49
5002	Same as 5001 exc btry pwr	2.79
5005	12 DIG 4 funct w/mem	2.99
MM5725	8 DIG 4 funct chain & dec	1.98
MM5736	18 pin 6 DIG 4 funct	4.45
MM5738	8 DIG 5 funct K & mem	5.35
MM5739	9 DIG 4 funct (btry sur)	5.35
MM5311	28 pin BCD 6 dig mux	4.45
MM5312	24 pin 1 pps BCD 4 dig mux	3.95
MM5313	28 pin 1 pps BCD 6 dig mux	4.45
MM5314	24 pin 6 dig mux	4.45
MM5316	40 pin alarm 4 dig	5.39

LED's

MV10B	Red TO 18	$ .22
MV50	Axial leads	.18
MV5020	Jumbo Vis. Red (Red Dome)	.22
	Jumbo Vis. Red (Clear Dome)	.22
ME4	Infra red diff. dome	.54
MAN1	Red 7 seg. .270"	2.19
MAN2	Red alpha num .32"	4.39
MAN4	Red 7 seq. .190"	1.95
MAN5	Green 7 seg. .270"	3.45
MAN6	.6" high solid seq.	4.25
MAN7	Red 7 seq. .270"	1.19
MAN3	Red 7 seg. .127" straight pins	.29
MAN8	Yellow 7 seq. .270"	3.45
MAN66	.6" high spaced seq.	3.75
MCT2	Opto-iso transistor	.61

MULTIPLE DISPLAYS

NSN33	3 digit .12" red led 12 pin fits IC skt.	$1.79
HP5082 7405	5 digit .11 led magn. lens com. cath	3.49
HP5082 7414	4 digit .11 LED magn. lens comm. cath.	$3.25
FNA37	9 digit 7 seg led RH dec clr. magn. lens	4.95
SP-425-09	9 digit .25" neon direct interface with MOS/LSI, 180 VDC, 7 seg	1.79

SHIFT REGISTERS

MM5013	1024 bit accum. dynamic	mDIP	$1.75
MM5016	500/512 bit dynamic	mDIP	1.59

DTL

930	$ .15	937	.15	949	.15
932	.15	944	.15	962	.15
936	.15	946	.15	963	.15

Data sheets on request
With order add $.30 for items less than $1.00 ea.

LINEAR CIRCUITS

300	Pos V Reg (super 723)	TO-5	$ .71
301	Hi Perf Op Amp	mDIP TO-5	.29
302	Volt follower	TO-5	.53
304	Neg V Reg	TO-5	.80
305	Pos V Reg	TO-5	.71
307	Op AMP (super 741)	mDIP TO-5	.26
308	Micro Pwr Op Amp	mDIP TO-5	.89
309K	5V 1A regulator	TO-3	1.35
310	V Follower Op Amp	mDIP	1.07
311	Hi perf V Comp	mDIP TO-5	.95
319	Hi Speed Dual Comp	DIP	1.13
320	Neg Reg 5.2, 12, 15	TO-3	1.19
322	Precision Timer	DIP	1.70
324	Quad Op Amp	DIP	1.52
339	Quad Comparator	DIP	1.58
340K	Pos V reg (5V, 6V, 8V, 12V, 15V, 18V, 24V)	TO-3	1.69
340T	Pos V reg (5V, 6V, 8V, 12V, 15V, 18V, 24V)	TO-220	1.49
372	AF-IF Strip detector	DIP	2.93
373	AM/FM/SSB Strip	DIP	.53
376	Pos V Reg	mDIP	2.42
377	2w Stereo amp	DIP	1.16
380	2w Audio Amp	DIP	1.13
380-8	.6w Audio Amp	mDIP	1.52
381	Lo Noise Dual preamp	DIP	1.52
382	Lo Noise Dual preamp	DIP	.71
550	Prec V Reg	DIP	.89
555	Timer	mDIP	.89
556A	Dual 555 Timer	DIP	1.49
560	Phase Locked Loop	DIP	2.48
562	Phase Locked Loop	DIP	2.48
565	Phase Locked Loop	DIP TO-5	2.38
566	Function Gen	mDIP TO-5	2.25
567	Tone Decoder	mDIP	2.66
709	Operational AMPL	TO-5 or DIP	.26
710	Hi Speed Volt Comp	DIP	.35
711	Dual Difference Compar	DIP	.26
723	V Reg	DIP	.62
739	Dual Hi Perf Op Amp	DIP	1.07
741	Comp Op AMP	mDIP TO-5	.32
747	Dual 741 Op Amp	DIP or TO-5	.71
748	Freq Adj 741	mDIP	.35
1304	FM Mulpx Stereo Demod	DIP	1.07
1307	FM Mulpx Stereo Demod	DIP	.74
1458	Dual Comp Op Amp	mDIP	.62
1800	Stereo multiplexer	DIP	2.48
LH2111	Dual LM 211 V Comp	DIP	1.70
3900	Quad Amplifier	DIP	.35
7524	Core Mem Sense AMPL	DIP	.71
8038	Voltage contr. osc.	DIP	4.25
8864	9 DIG Led Cath Drvr	DIP	2.25
75150	Dual Line Driver	DIP	1.75
75451	Dual Perepheral Driver	mDIP	.35
75452	Dual Peripheral Driver	mDIP	.35
75453	(351) Dual Periph Driver	mDIP	.35
75491	Quad Seq Driver for LED	DIP	.71
75492	Hex Digit driver	DIP	.80

Satisfaction guaranteed. Shipment will be made via first class mail in U.S., Canada and Mexico within 3 days from receipt of order. Add $.50 to cover shipping and handling for orders under $25.00. Minimum order $5.00. California residents add sales tax.

INTERNATIONAL ELECTRONICS UNLIMITED



P.O. BOX 1708/ MONTEREY, CA. 93940 USA
PHONE **(408) 659-3171**

Carl Helmers

Benchmarks, Standards, etc.

I just described what I think is a reasonable version of the state of the art in small systems computing at the low end in price. Now, what can you expect out of the supplier of small systems products so that you can implement the benchmark capability? Well, for one thing, sooner or later every supplier will want to demonstrate this type of system with an editing capability. Note with care that I did not specify one iota of detail about the actual editor programs themselves which implement the capability. I've hardly limited the software people (or lack of same) at the kit companies to a particular implementation. There will be plenty of room for variations on the basic editor concept from supplier to supplier, in terms of meeting the benchmark requirement.

The only supplier of kit computers who comes close to demonstrating the under $1000 benchmark to my satisfaction is Scelbi Computer Consulting in Milford, Conn. Their 8008 oriented system has reasonably priced software listed in their catalog for a tape editor program, and the peripherals/memory products they list look sufficient for this capability. (MITS: I get the impression that from your prices and available printed data, your library of programs is oriented to the larger systems which implement your BASIC interpreter. This puts them out of the class of under $1000 computer systems, although your new 6800 system may be in the small computer benchmark classification of this editorial.)

Standardization is a must for the growth of this industry – both for the users of the equipment and the manufacturers.

As manufacturers send me evidence (documentation, systems summaries, program listings, etc.) for the small systems benchmark capability, I'll report on them either in the editorials of BYTE or in review articles commissioned for that purpose. Readers can help prod the software departments (or lack of same) at the various kit manufacturers by writing them to urge demonstration of both the benchmark capability and to urge "bundling" of that valuable systems software – the editor – as part of the package of items which comes with the computer.

Comments on Kit Documentation: An Open Letter to Manufacturers

What should readers expect from a kit manufacturer? BYTE is certainly looking into the various kit products with the reviewers' soldering irons, pens and typewriters in hand, so we can present a critical evaluation of the ease (or lack thereof) of assembly of the products. Just so you won't have it sprung on you without warning, I thought a few comments on assembly manual standards for reviewing kits might help, and it will give our readers an idea of what to look for when they buy these products. For starters, let's concentrate on the first documents the kit builder has to come across – the system's assembly and hardware checkout documentation.

Manufacturers, be on your guard here! Microcomputers are complicated pieces of equipment which cannot tolerate mistakes in assembly if the result is to be a functional unit. Further, since there is a finite possibility of bad integrated circuits getting through your quality control procedures (if any), you had best make sure you can count upon your customers to diagnose any problems which might arise. While you can assume an above average intelligence on the part of your customers – the readers of BYTE – you cannot assume a complete familiarity with all the details of your system. The burden of clear and understandable presentation lies upon your shoulders. The days of the two page description of some digital logic circuit being passed off as an assembly manual are over when it comes to microcomputers. Kit computer builders do not need the Heathkit overkill approach to assembly procedures, but they do need more than a skeleton of a manual. There is a balanced middle ground between the extreme of minute detail on one hand, and a bare circuit diagram with parts list on the other. Here are some of my thoughts on what a good assembly and checkout document might contain.

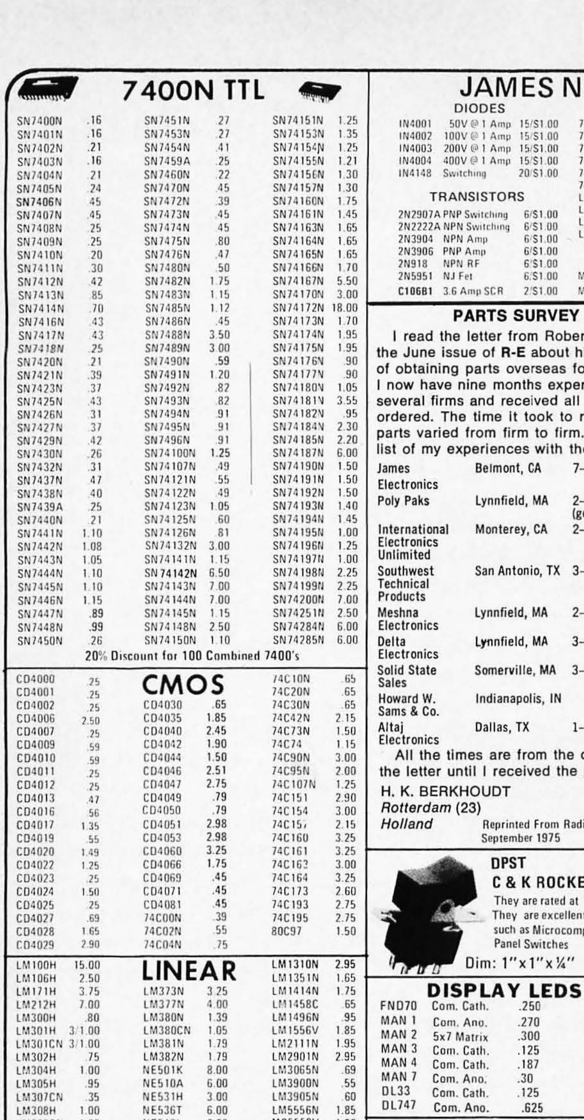
System Summary: A description of the overall design of your product, and where the particular unit being assembled fits into that design.

Complete Schematics: The need for these goes without saying.

Parts Lists: Specify what you think you ship, and stick to it as closely as possible. The user can check it off and pick up errors.

Recommended Assembly Sequences: Give a fairly detailed sequence of assembly, more than "stuff PC boards," "solder" then "checkout." As a manufacturer you should assemble the kit at least once yourself while taking notes on paper or magnetic tape. In any given mechanical and electrical assembly design, there is bound to be an optimal sequence of steps – being close to it you should be able to give a good approximation of the optimum. The assembly sequence should be specified with sufficient detail – pictures where necessary – so that an intelligent reader can reproduce it.

Checkout Sequences And Self-Test: Thought should be given to the sequence of procedures needed to verify the operation of the kit once it is assembled. Do not under any circumstances leave your checkout procedures to a strictly "nominal" course. I can predict that sooner or later some of your customers will find common glitches in assembly. Give a few side branches in the checkout procedures for commonly occurring problems. "Ah," you protest, "but I don't



7400N TTL

SN7400N	.16	SN7451N	.27	SN74151N	1.25
SN7401N	.16	SN7453N	.27	SN74153N	1.35
SN7402N	.21	SN7454N	.41	SN74154N	1.25
SN7403N	.16	SN7459A	.25	SN74155N	1.21
SN7404N	.21	SN7460N	.22	SN74156N	1.30
SN7405N	.24	SN7470N	.45	SN74157N	1.30
SN7406N	.45	SN7472N	.39	SN74160N	1.75
SN7407N	.45	SN7473N	.45	SN74161N	1.45
SN7408N	.25	SN7474N	.45	SN74163N	1.65
SN7409N	.25	SN7475N	.80	SN74164N	1.65
SN7410N	.20	SN7476N	.47	SN74165N	1.65
SN7411N	.30	SN7480N	.50	SN74166N	1.70
SN7412N	.42	SN7482N	1.75	SN74167N	5.50
SN7413N	.85	SN7483N	1.15	SN74170N	3.00
SN7414N	.70	SN7485N	1.12	SN74172N	18.00
SN7416N	.43	SN7486N	.45	SN74173N	1.70
SN7417N	.43	SN7488N	3.50	SN74174N	1.95
SN7418N	.25	SN7489N	3.00	SN74175N	1.95
SN7420N	.21	SN7490N	.59	SN74176N	.90
SN7421N	.39	SN7491N	1.20	SN74177N	.90
SN7423N	.37	SN7492N	.82	SN74180N	1.05
SN7425N	.43	SN7493N	.82	SN74181N	3.55
SN7426N	.31	SN7494N	.91	SN74182N	.95
SN7427N	.37	SN7495N	.91	SN74184N	2.30
SN7429N	.42	SN7496N	.91	SN74185N	2.20
SN7430N	.26	SN74100N	1.25	SN74187N	6.00
SN7432N	.31	SN74107N	.49	SN74190N	1.50
SN7437N	.47	SN74121N	.55	SN74191N	1.50
SN7438N	.40	SN74122N	.49	SN74192N	1.50
SN7439A	.25	SN74123N	1.05	SN74193N	1.40
SN7440N	.21	SN74125N	.60	SN74194N	1.45
SN7441N	1.10	SN74126N	.81	SN74195N	1.00
SN7442N	1.08	SN74132N	3.00	SN74196N	1.25
SN7443N	1.05	SN74141N	1.15	SN74197N	1.00
SN7444N	1.10	SN74142N	6.50	SN74198N	2.25
SN7445N	1.10	SN74143N	7.00	SN74199N	2.25
SN7446N	1.15	SN74144N	7.00	SN74200N	7.00
SN7447N	.89	SN74145N	1.15	SN74251N	2.50
SN7448N	.99	SN74148N	2.50	SN74284N	6.00
SN7450N	.26	SN74150N	1.10	SN74285N	6.00

20% Discount for 100 Combined 7400's

CMOS

CD4000	.25			74C10N	.65
CD4001	.25			74C20N	.65
CD4002	.25	CD4030	.65	74C30N	.65
CD4006	2.50	CD4035	1.85	74C42N	2.15
CD4007	.25	CD4040	2.45	74C73N	1.50
CD4009	.59	CD4042	1.90	74C74	1.15
CD4010	.59	CD4044	1.50	74C90N	3.00
CD4011	.25	CD4046	2.51	74C95N	2.00
CD4012	.25	CD4047	2.75	74C107N	1.25
CD4013	.47	CD4049	.79	74C151	2.90
CD4016	.56	CD4050	.79	74C154	3.00
CD4017	1.35	CD4051	2.98	74C157	2.15
CD4019	.55	CD4053	2.98	74C160	3.25
CD4020	1.49	CD4060	3.25	74C161	3.25
CD4022	1.25	CD4066	1.75	74C163	3.00
CD4023	.25	CD4069	.45	74C164	3.25
CD4024	1.50	CD4071	.45	74C173	2.60
CD4025	.25	CD4081	.45	74C193	2.75
CD4027	.69	74C00N	.39	74C195	2.75
CD4028	1.65	74C02N	.55	80C97	1.50
CD4029	2.90	74C04N	.75		

LINEAR

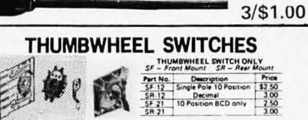
LM100H	15.00			LM1310N	2.95
LM106H	2.50			LM1351N	1.65
LM171H	3.75	LM373N	3.25	LM1414N	1.75
LM212H	7.00	LM377N	4.00	LM1458C	.65
LM300H	.80	LM380N	1.39	LM1496N	.95
LM301H	3/1.00	LM380CN	1.05	LM1556V	1.85
LM301CN	3/1.00	LM381N	1.79	LM2111N	1.95
LM302H	.75	LM382N	1.79	LM2901N	2.95
LM304H	1.00	NE501K	8.00	LM3065N	.69
LM305H	.95	NE510A	6.00	LM3900N	.55
LM307CN	.35	NE531H	3.00	LM3905N	.60
LM308H	1.00	NE536T	6.00	LM5556N	1.85
LM308CN	1.00	NE540L	6.00	MC5558V	1.00
LM309H	1.10	NE550N	.79	LM7525N	.90
LM309K	1.25	NE553	2.50	LM7528N	2.20
LM310CN	1.15	NE555V	.49	LM7534N	2.20
LM311H	.90	NE565H	.99	LM7535N	1.25
LM311N	.90	NE565N	1.25	8038B	4.95
LM318CN	1.50	NE566CN	1.95	LM75450	.49
LM319N	1.30	NE567H	1.25	75451CN	.39
LM319D	9.00	NE567V	1.50	75452CN	.39
LM320K-5	1.35	LM703CN	.45	75453CN	.39
LM320K-5.2	1.35	LM709H	.29	75454CN	.39
LM320K-12	1.35	LM709N	.29	75491CN	.79
LM320K-15	1.35	LM710N	.79	75492CN	.89
LM323K-5	10.50	LM711N	.39	75494CN	.89
LM324N	1.80	LM723N	.55	**RCA LINEAR**	
LM339N	1.70	LM723H	.55	CA3013	1.70
LM340K-5	1.95	LM733N	1.00	CA3023	2.15
LM340K-12	1.95	LM739N	1.29	CA3035	2.25
LM340K-15	1.95	LM741CH	3/1.00	CA3039	1.35
LM340K-24	1.95	LM741CN	3/1.00	CA3046	1.15
LM340To-5	1.75	LM741-14N	.39	CA3059	2.46
LM340To-6	1.75	LM747H	.79	CA3060	2.80
LM340To-12	1.75	LM747N	.79	CA3080	.85
LM340To-15	1.75	LM748H	.39	CA3083	1.60
LM340To-24	1.75	LM748N	.39	CA3086	.59
LM350N	1.00	LM1303N	.90	CA3089	3.25
LM351CN	.65	LM1304N	1.19	CA3091	8.25
LM370N	1.15	LM1305N	1.40	CA3123	1.85
LM370H	1.15	LM1307N	.85	CA3600	1.75

DATA HANDBOOKS

7400	Pin-out & Description of 5400/7400 ICS	$2.95
CMOS	Pin-out & Description of 4000 Series ICS	$2.95
LINEAR	Pin-out & Functional Description of Linear Circuits	$2.95

6' POWER SUPPLY CORDS ....

2 CONDUCTORS



SPECIAL 3/$1.00

THUMBWHEEL SWITCHES



Snap together – No Hardware

THUMBWHEEL SWITCH ONLY
SF – Front Mount SR – Rear Mount

Part No.	Description	Price
SF-12	Single Pole 10 Position	$2.50
SR-12	Decimal	3.00
SF-21	10 Position BCD only	2.50
SR-21		3.00

SERIES SF Front Mount Assembly ($11.70)



Note: Picture to the left has the following 4 ea. SF-12, 1 ea. SF-EP, 1 ea. SF-BB, 1 ea. SF-DP and 1 ea. SF-HB.

ACCESSORIES

Part No.	Description	Price
SF-EP	End Plate (Pair)	.50
SF-DP	Divider Plate (each)	.40
SF-BB	Blank Body (each)	.40
SF-HB	Half Body (each)	.40

SERIES SR Rear Mount Assembly ($13.70)

Note: Picture to the left has the following 4 ea. SR-12, 1 ea. SR-EP, 1 ea. SR-BB, 1 ea. SR-DP and 1 ea. SR-HB.

ACCESSORIES

Part No.	Description	Price
SR-EP	End Plates (pair)	$ .50
SR-DP	Divider Plate (each)	.40
SR-BB	Blank Body (each)	.40
SR-HB	Half Body (each)	.40

Ordering: Order desired switch or switches and add necessary accessories for your particular application.

JAMES NOVEMBER SPECIALS

DIODES

IN4001	50V @ 1 Amp	15/$1.00
IN4002	100V @ 1 Amp	15/$1.00
IN4003	200V @ 1 Amp	15/$1.00
IN4004	400V @ 1 Amp	15/$1.00
IN4148	Switching	20/$1.00

TRANSISTORS

2N2907A	PNP Switching	6/$1.00
2N2222A	NPN Switching	6/$1.00
2N3904	NPN Amp	6/$1.00
2N3906	PNP Amp	6/$1.00
2N918	NPN RF	6/$1.00
2N5951	NJ Fet	6/$1.00
C106B1	3.6 Amp SCR	2/$1.00

TTL/LINEAR

7400	Gate	7/$1.00
7447	Decoder	.79
7490	Counter	.49
74100	8 Bit Latch	.99
74154	Decoder	.99
74197	Counter	.75
LM301H–LM741H		4/$1.00
LM324N–Quad 741		1.19
LM309H	5V Reg TO-5	.79
LM309K	5V Reg TO-3	.99

DYNAMIC SHIFT REGISTERS

MM500H MM504H MM510H 2 FOR
MM503H MM506H MM5016H $1.00

CLOCK & CALC. CHIPS

MM5311N	6 Digit	$2.95
MM5312N	4 Digit	2.95
MM5313N	6 Digit	2.95
MM5314N	6 Digit	2.95
MM5316N	6 Digit Alarm	3.95
MM5725N	8 Digit-4 Funt.	1.98
MM5736N	6 Digit-4 Funt.	3.95
MM5738N	8 Digit-5 Funt.	4.95

26 Awg RIBBON CABLE

1 Ft. Minimum	1-9 ft.	10 ft.
4 Cond.	.29 ft.	.25 ft.
8 Cond.	.59 ft.	.49 ft.
16 Cond.	.89 ft.	.79 ft.
32 Cond.	1.89 ft.	1.69 ft.

PARTS SURVEY

I read the letter from Robert Briner in the June issue of **R-E** about his problem of obtaining parts overseas for projects. I now have nine months experience with several firms and received all the parts I ordered. The time it took to receive the parts varied from firm to firm. Here is a list of my experiences with the firms.

James Electronics	Belmont, CA	7–10 days
Poly Paks	Lynnfield, MA	2–3 weeks (getting better)
International Electronics Unlimited	Monterey, CA	2–3 weeks
Southwest Technical Products	San Antonio, TX	3–4 weeks
Meshna Electronics	Lynnfield, MA	2–3 weeks
Delta Electronics	Lynnfield, MA	3–4 weeks
Solid State Sales	Somerville, MA	3–4 weeks
Howard W. Sams & Co.	Indianapolis, IN	7 weeks
Altaj Electronics	Dallas, TX	1–1½ months

All the times are from the day I wrote the letter until I received the parts.

H. K. BERKHOUDT
Rotterdam (23)
Holland

Reprinted From Radio & Electronics September 1975

HP-5082-7300

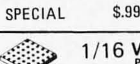
HP-5082-7300
.3" Dot Matrix type numeric readouts with decoder/driver/latch built on the chip. Only 8 bins (BCD in, DP, Latch, +5v, ground).
HP 5082-7300 $5.95

HP 5082-7304 + 1 version of 7300 4.95

1¼" x 1½" XFMERS P.C. Mount

That were designed for clock type applications. 110Vac primary @60Hz. Secondaries: 8-10Vac @ 300mA-500mA
50Vac @ 30mA-50mA
Excellent for miniature power supplies & gas discharge displays

SPECIAL $.99

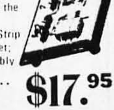


1/16 VECTOR BOARD

P-Pattern
0.1" Hole Spacing

PHENOLIC	64P44 062XXXP	4.50	6.50	1.72	1.54
	169P44 062XXXP	4.50	17.00	3.69	3.32
EPOXY GLASS	64P44 062	4.50	6.50	2.07	1.86
	84P44 062	4.50	8.50	2.56	2.31
	169P44 062	4.50	17.00	5.04	4.53
	169P84 062	8.50	17.00	9.23	8.26
EPOXY GLASS COPPER CLAD	169P44 062C1	4.50	17.00	6.80	6.12

PROTO BOARD-100



Here's a low cost, big 10 IC capacity breadboard kit with all the quality of OT Sockets and the best of the Proto Board series ... complete down to the last nut, bolt and screw. Includes 2 QT-35S Sockets; 1 QT-35B Bus Strip 2 5-way binding posts; 4 rubber feet; screws, nuts, bolts; and easy assembly instructions.

COMPLETE KIT ... *Special $17.95



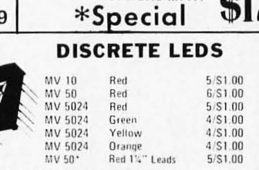
DPST C & K ROCKER SWITCH

They are rated at 125 Vac @ 5A. They are excellent in application such as Microcomputer Panel Switches

Dim: 1"x1"x¼" $0.69

DISPLAY LEDS

FND70	Com. Cath.	.250	.59
MAN 1	Com. Ano.	.270	$1.95
MAN 2	5x7 Matrix	.300	3.95
MAN 3	Com. Cath.	.125	.39
MAN 4	Com. Cath.	.187	1.95
MAN 7	Com. Ano.	.30	1.50
DL33	Com. Cath.	.125	1.95
DL747	Com. Ano.	.625	1.95



DISCRETE LEDS

MV 10	Red	5/$1.00
MV 50	Red	6/$1.00
MV 5024	Red	5/$1.00
MV 5024	Green	4/$1.00
MV 5024	Yellow	4/$1.00
MV 5024	Orange	4/$1.00
MV 50*	Red 1¼" Leads	5/$1.00

IC SOLDERTAIL – LOW PROFILE (TIN) SOCKETS

	1-24	25-49	50-100		1-24	25-49	50-100
8 pin	$.17	.16	.15	24 pin	$ .38	.37	.36
14 pin	.20	.19	.18	28 pin	.45	.44	.43
16 pin	.22	.23	.22	36 pin	.60	.59	.58
18 pin	.29	.28	.27	40 pin	.63	.62	.61
22 pin	.36	.37	.36				

SOLDERTAIL STANDARD (GOLD)

8 pin	$.30	.27	.24	24 pin	$ .70	.63	.57
14 pin	.35	.32	.29	28 pin	1.10	1.00	.90
16 pin	.38	.35	.32	36 pin	1.55	1.40	1.26
18 pin	.52	.47	.43	40 pin	1.75	1.59	1.45

WIRE WRAP SOCKETS (GOLD) LEVEL #3

10 pin	$.45	.41	.37	24 pin	$1.05	.95	.85
14 pin	.39	.38	.37	28 pin	1.40	1.25	1.10
16 pin	.43	.42	.41	36 pin	1.59	1.45	1.30
18 pin	.75	.68	.62	40 pin	1.75	1.55	1.40

50 PCS. RESISTOR ASSORTMENTS $1.75 PER ASST.

ASST. 1	5 ea:	10 OHM 12 OHM 15 OHM 18 OHM 22 OHM 27 OHM 33 OHM 39 OHM 47 OHM 56 OHM	1/4 WATT 5% = 50 PCS.
ASST. 2	5 ea:	68 OHM 82 OHM 100 OHM 120 OHM 150 OHM 180 OHM 220 OHM 270 OHM 330 OHM 390 OHM	1/4 WATT 5% = 50 PCS.
ASST. 3	5 ea:	470 OHM 560 OHM 680 OHM 820 OHM 1K 1.2K 1.5K 1.8K 2.2K 2.7K	1/4 WATT 5% = 50 PCS.
ASST. 4	5 ea:	3.3K 3.9K 4.7K 5.6K 6.8K 8.2K 10K 12K 15K 18K	1/4 WATT 5% = 50 PCS.
ASST. 5	5 ea:	22K 27K 33K 39K 47K 56K 68K 82K 100K 120K	1/4 WATT 5% = 50 PCS.
ASST. 6	5 ea:	150K 180K 220K 270K 330K 390K 470K 560K 680K 820K	1/4 WATT 5% = 50 PCS.
ASST. 7	5 ea:	1M 1.2M 1.5M 1.8M 2.2M 2.7M 3.3M 3.9M 4.7M 5.6M	1/4 WATT 5% = 50 PCS.

PRIME INTEGRATED CIRCUIT ASSORTMENTS

ASST. 8	3 ea.	SN7400 7401 7402 7403 7404 SN7410 7430 7438 7440 7472	SSI/TTL	$5.95 ASST.
ASST. 9	2 ea.	SN7447 7490 7491 74100 74121 SN74145 74175 74180 74191 74193	MSI/TTL	$9.95 ASST.
ASST. 10	2 ea.	CD4001 4002 4011 4012 4013 CD4016 4017 4019 4023 4030	CMOS	$7.95 ASST.
ASST. 11	2 ea.	LM301T 301N 302T 307T 309K LM311T 565T 567T 741T 741N	LINEAR	$10.95 ASST.

Satisfaction Guaranteed. $5.00 Min. Order. U.S. Funds.
California Residents — Add 6% Sales Tax
Write for FREE 1975F Catalog — Data Sheets .25¢ each



P.O. BOX 822, BELMONT, CA. 94002
PHONE ORDERS — (415) 592-8097

MICROPROCESSOR COMPONENTS

8008 – $19.95 2102 – $2.95

CPU'S				RAM'S		
8008	8 bit CPU	$19.95	1101	256x1	Static	$ 2.25
8080	8 bit Super 8008	149.95	1103	1Kx1	Dynamic	2.95
4040	4 bit 8080	39.95	2102	1Kx1	Static	2.95
	UARTS		2107	4Kx1	Dynamic	19.95
AY-5-1013		6.95	7010	1Kx1	Mnmos	29.95
	PROM'S		7489	16x4	Static	2.49
8223	32x8 Prom	3.00	8101	1Kx1	Static	7.95
1702A	2K E Prom	17.95	8111	1Kx1	Static	7.95
5203Q	2K E Prom	14.95	8599	16x4	Tristate	3.49
74S287	1Kx1 Prom	7.95	91L02	1Kx1	Low Power	3.95
	SR'S		74200	256x1	Static	6.95
2401	2K DSR	9.95	74LS670	4x4	Register	3.95
2533	1K SSR	7.95	93410	256x1	Static	2.95

91L02 1K Static RAM Direct Replacement for 2102-1 with 40% Less Current Drawn. **3.95**

Vector General Purpose Logic CARD Board 14.95
*Very High Noise Immunity * Holds 12 ea. 14 pin DIPS
*44 pin Edge Connection

THE KILOBYTE RAM CARD Per Kit $49.95
*Complete 1Kx8 Memory * High Noise Immunity Components
*Single 5v supply *500NS Access Time & Kit includes sockets, ICS & Board

FREE $12.00 Value With each $75 order of Microprocessor components, get a one year subscription to BYTE the magazine for computer PHREAKS FREE FREE FREE

KITS EXAR ICS

XR-2206KA SPECIAL $17.95
Includes monolithic function generator IC, PC board, and assembly instruction manual.

XR-2206KB SPECIAL $27.95
Same as XR-2206KA above and includes external components for PC board.

TIMERS

XR-555CP	SPECIAL	$ .69
XR-320P		1.55
XR-556CP		1.85
XR-2556CP		3.20
XR-2240CP	SPECIAL	3.25

PHASE LOCKED LOOPS

XR-210		5.20
XR-215		6.60
XR-567CP		1.95
XR-2567CP	SPECIAL	2.99

STEREO DECODERS

XR-1310P	$3.20
XR-1310EP	3.20
XR-1800P	3.20

WAVEFORM GENERATORS

XR-205		8.40
XR-2206CP	SPECIAL	4.49
XR-2207CP		3.85

MISCELLANEOUS

XR-2211CP	6.70
XR-2261	3.79

Special Requested Items

RC4194	Dual Track V Reg	$5.95	N8T97	$3.00	2533	$7.95
RC4195	±15V Track Reg	3.25	4024P	2.25	8263	5.95
F9368	Decoder	3.95	2513	11.00	8267	2.75
LD110/111	DVM Chip Set	28.00	2518	7.00	8288	1.15
CA3130	Super CMOS Op Amp	1.49	2524	3.50	8826	3.00
MC1408L7	A/D	9.95	2525	6.00	8880	1.35
F3341	FIFO	8.95	2527	5.00	7497	5.00

ZENERS—DIODES—RECTIFIERS

TYPE	VOLTS	W	PRICE	TYPE	VOLTS	W	PRICE
IN746	3.3	400m	4/1.00	IN4003	200 PIV	1 AMP	.10
IN751A	5.1	400m	4/1.00	IN4004	400 PIV	1 AMP	.10
IN752	5.6	400m	4/1.00	IN3600	50	200m	6/1.00
IN753	6.2	400m	4/1.00	IN4148	75	10m	15/1.00
IN754	6.8	400m	4/1.00	IN4154	35	10m	12/1.00
IN965B	15	400m	4/1.00	IN4734	5.6	1w	.28
IN5232	5.6	500m	.28	IN4735	6.2	1w	.28
IN5234	6.2	500m	.28	IN4736	6.8	1w	.28
IN5235	6.8	500m	.28	IN4738	8.2	1w	.28
IN5236	7.5	500m	.28	IN4742	12	1w	.28
IN456	25	40m	6/1.00	IN4744	15	1w	.28
IN458	150	7m	6/1.00	IN1183	50 PIV	35 AMP	1.60
IN485A	180	10m	5/1.00	IN1184	100 PIV	35 AMP	1.70
IN4001	50 PIV	1 AMP	.09	IN1186	200 PIV	35 AMP	1.80
IN4002	100 PIV	1 AMP	.10	IN1188	400 PIV	35 AMP	3.00

TRANSISTORS

MPS-A05	5/1.00			PN4249	4/$1.00
MPS-A06	5/$1.00	PN3567	3/$1.00	PN4250	4/$1.00
2N2219A	3/$1.00	PN3568	4/$1.00	2N4400	4/$1.00
2N2221	4/$1.00	PN3569	4/$1.00	2N4401	4/$1.00
2N2222A	5/$1.00	2N3704	5/$1.00	2N4402	4/$1.00
2N2369	5/$1.00	2N3705	5/$1.00	2N4403	4/$1.00
2N2369A	4/$1.00	2N3700	5/$1.00	2N4409	5/$1.00
FN2415	5/$1.00	2N3707	5/$1.00	2N5086	4/$1.00
2N2484	4/$1.00	2N3711	5/$1.00	2N5087	4/$1.00
2N2906A	4/$1.00	2N3724	$.65	2N5088	4/$1.00
2N2907A	5/$1.00	2N3725	$1.00	2N5089	4/$1.00
2N2925	5/$1.00	2N3903	5/$1.00	2N5129	5/$1.00
2N3053	2/$1.00	2N3904	4/$1.00	2N5138	5/$1.00
2N3055	$.89	2N3905	4/$1.00	2N5139	5/$1.00
MJE3055	$.89	2N3906	4/$1.00	2N5209	5/$1.00
2N3392	5/$1.00	2N4013	3/$1.00	2N5951	5/$1.00
2N3398	5/$1.00	2N4014	3/$1.00	C106B1 SCR	2/$1.00

CAPACITOR CORNER

50 VOLT CERAMIC DISC CAPACITORS

	1-9	10-49	50-100		1-9	10-49	50-100
10pf	.05	.04	.03	.001	.05	.04	.035
22 pf	.05	.04	.03	.0047	.05	.04	.035
47 pf	.05	.04	.03	.01	.05	.04	.035
100 pf	.05	.04	.03	.022	.06	.05	.04
220 pf	.05	.04	.03	.047	.06	.05	.04
470 pf	.05	.04	.035	.1	.12	.09	.075

100 VOLT MYLAR FILM CAPACITORS

.001mf	.12	.10	.07	.022mf	.13	.11	.08
.0022	.12	.10	.07	.047mf	.21	.17	.13
.0047mf	.12	.10	.07	.1mf	.27	.23	.17
.01mf	.12	.10	.07	.22mf	.33	.27	.22

±20% DIPPED TANTALUMS (SOLID) CAPACITORS

.1	35V	.28	.23	.17	1.5	35V	.30	.26	.21
.15	35V	.28	.23	.17	2.2	25V	.31	.27	.22
.22	35V	.28	.23	.17	3.3	25V	.31	.27	.22
.33	35V	.28	.23	.17	4.7	25V	.32	.28	.23
.47	35V	.28	.23	.17	6.8	25V	.36	.31	.25
.68	35V	.28	.23	.17	10	25V	.40	.35	.29
1.0	35V	.28	.23	.17	15	25V	.63	.50	.40

MINIATURE ALUMINUM ELECTROLYTIC CAPACITORS

Axial Lead					Radial Lead				
.47	50	.15	.13	.10	.47	25	.15	.13	.10
1	50	.16	.14	.11	.47	50	.16	.14	.11
3.3	50	.15	.13	.10	1	16	.15	.13	.10
4.7	25	.16	.14	.12	1	25	.16	.14	.11
10	25	.15	.13	.10	1	50	.16	.14	.11
10	50	.16	.14	.12	4.7	16	.15	.13	.10
22	25	.17	.15	.12	4.7	25	.15	.13	.10
22	50	.24	.20	.18	4.7	50	.16	.14	.11
47	25	.19	.17	.15	10	16	.14	.12	.09
47	50	.25	.21	.19	10	25	.15	.13	.10
100	25	.24	.20	.18	10	50	.16	.14	.12
100	50	.35	.30	.28	47	50	.24	.21	.19
220	25	.32	.28	.25	100	16	.19	.15	.14
220	50	.45	.41	.38	100	25	.24	.20	.18
470	25	.33	.29	.27	100	50	.35	.30	.28
1000	16	.55	.50	.45	220	16	.23	.17	.16
2200	16	.70	.62	.55	470	25	.31	.28	.26

know what the problems are!" Well, why not institute an in-house contest among your engineers called BUGDEBUG: Have one fellow go to a prototype and intentionally introduce a bug, then have his compatriot – without advance knowledge – get that machine and figure out what is wrong while taking notes on his methods of deducing the error.

Use of the Computer as a Debugging Tool: Above all, concentrate on getting the CPU up first in your procedures, so that it can be used as a debugging tool by the fellow who has no oscilloscope. Think about the computer as a debugger, and burn some hardware debugging routines into your ROMs.

I present these thoughts as a constructive input to all manufacturers – and as information for readers to use in the evaluation of your products. I predict that the manufacturers who do the best job documenting their kits will be the ones who prosper the most in the long run.

Standardization

As I mentioned in describing the small systems benchmark, the idea of a home computer system is best implemented with modularity. This brings up the question of standardization of interfaces, since one of the most obvious areas of modularity is in the choice of peripheral devices for a system. What are standards, and whom do they serve?

BYTE will editorially support any equipment that meets industry standards, however those standards are defined.

A standard is an agreement – de facto or by explicit choice of participants – upon a common set of parameters for some aspect of technology. In the home computer field, the de facto standard of character data is adopted without question from the computer industry at large – ASCII (Baudot Teletype interfaces use obsolete equipment and are viewed as an exception to be specially programmed). There is, however, a strong need for additional standardization in two areas, one of which I'll talk about in some detail now. These are the areas of peripheral interconnection to processors and the audio cassette data interchange standard. Work upon the interface standardization problem has already begun under the leadership of Bill Godbout in Oakland, Calif. Bill has been in and around the computer industry for a long time (he worked for IBM when IBM was anything but a giant in the field) and appreciates well the shot in the arm competition can give the development of a new area. By having plug standards, we can avoid arbitrary restrictions on entry into the field, thereby expanding the choices users have and giving incentives to the innovative and creative producers of personal computing products. With a peripheral standard for the industry, the inventor of "El Neato Peripheral #367" only has to worry about one interface, and can enhance the value of Bill Godbout's computer systems, as well as Sphere, MITS, SWTPC, RGS, Scelbi and yet unheard of products.

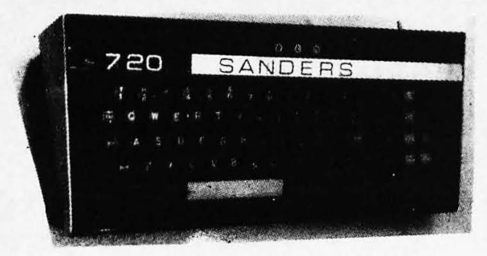
In California, there already have been several informal standards meetings among industry representatives, according to Bill in a phone conversation recently . . . a trend which we at BYTE want to see continuing.

As a step in a second direction – audio cassette standards – BYTE is sponsoring an audio cassette standardization conference on November 7 and 8 in Kansas City, Kansas (see the announcement in this issue). The purpose of this conference is to act as an information exchange among the various manufacturers and users, and to hear arguments on the relative merits of various systems for audio recording of digital data. The main argument in favor of the audio standard concept – whatever technically competent system or systems are chosen – is so that the various users of small computers can freely communicate with one another by passing data tapes back and forth. We need a universal audio standard so that Joe Smyly in Podunk Hollow, Kentucky, can record his thoughts on tape and send them to his friend Fred Oberheimer in San Francisco, where Fred can display the message on his own computer's data screens or hard copy outputs.

Once a standard has been agreed upon you may be sure that BYTE will do all it can to encourage manufacturers to adhere to this standard and to publish articles explaining the systems involved, the hardware which will work with it, and software involved. Should some alternatives be proposed which seem of value, BYTE will encourage their investigation and evaluation. But once a standard has been set, it will take some powerful improvements to warrant basic changes.

Carl Helmers
Editor, BYTE

COMPUTER–DATA INPUT KEYBOARDS



B5283



B5199

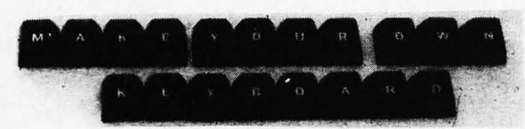
ASCII encoded keyboard. In its own enclosure. Originaly used in SANDERS ASSOCIATES 720 Terminal System. In like new condition. Usefull for any project requiring an ASCii encoded keyboard. 50 Alpha Numeric keys plus 11 computer symbols

STOCK NO.B5283 keyboard $35.00 2/65.00

MICRO—SWITCH (Honeywell) 8 bit binary coded board. 56 keys, alpha - meric and computer symbols Built in TTL decoder. New in factory cartons. A beautiful keyboard.

STOCK NO.B5199 Microswitch keyboard. $45.00 2/80.00

KEYTOPS & SWITCHES TO MAKE YOUR OWN KEYBOARD



We have a large selection of KEYTOPS and SWITCHES, made by RAYTHEON CO. The keytops come in black, grey and white, with contrasting legends. The switches mate with the tops, and are magnetic reed switches. The following combinations are available:

54 key typewriter set, keys only,	black	K9276	2.95
54 key typewriter set, keys only,	grey	K9278	2.95
54 key TTY set, no symbols	white	K9279	2.95
54 key TTY set, with symbols	white	K9282	2.95
54 key set, keys & switches	black	K9288	30.00
54 key set, keys & switches	grey	K9290	30.00
54 TTY set, no symbols keys & Sw.	White	K9291	30.00
54 TTY set, with symbols, keys & Sw.	white	K9291	30.00
11 Key Numeric set. Keys only	Black	K9283	1.50
11 Key Numeric set, Keys only	Grey	K9284	1.50
11 Key Numeric set, Keys only	White	K9295	1.50
12 Key numeric set, Keys only	white	K9286	1.50
11 Key Num.set, keys & switches	Black	K9293	7.00
11 Key Num. set, keys & switches	Grey	K9294	7.00
11 Key Num. set, keys & switches	White	K9295	7.00
12 Key Num. set, keys & switches	white	K9296	7.50
Blank key 1½ keys wide	white	K9297A	3/.25
Blank key 2 keys wide	white	K9297B	3/.25
K9297A with switch	white	K9298A	3/2.00
K9297B with switch	white	K9298B	3/2.00

MINIATURE 7 SEGMENT READOUT

Miniature 7 segment LED readout (EXITON XMN 101). Displays all numbers and 9 letters. O.D. 5/16"x 1/4" Display is .12 ". SPECIAL FOR THIS ISSUE ONLY

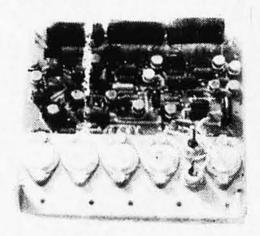
STOCK NO.B5173 with data sheet .50 ea. 5/2.00

TRANSFORMERS

Computer projects need power supplies. Finding the right power transformer can be a problem. We have one of the largest and most diversified stocks of power transformers in the country. Below we list some representative items in our inventory. Our catalog, free on request lists many more.

36 V. @ 1.0 A. ct, & 6.3 V @ 200 ma.	3 lbs.	B9313	3.50 2/6.00
70 V. @ 1.5 A. ct, & 6.3 V @ 500 ma.	6 lb.	B9314	6.50 2/12.00
90 V. @ 2.0 A. ct, & 6.3 V @ 1.5 A.	8½ lb.	B9315	9.95 2/19.00
50 V.=@ 1.5 A. ct, & 6.3V @500 ma.	6 lb.	B9316	6.50 2/12.00
26 V. @ 1.0 A. ct. &6.3 V. @ 500 ma.	3 lb.	B9318	3.75 2/7.00
38 V. @ 1.5 A. ct. & 6.0 V.@ 500 ma.	2 lb.	B9319	6.95 2/13.00
350 V. @ 35 ma. ct. & 6.3 V. @ 2.7 A	2 lb.	B9321	3.50 2/6.00
70 V. @ 1.5 A. ct. & 6.3 V @ 1.5 A.	7Lb.	B9322	6.75 2/13.00
35 V. @ 6.0 Ct. & 10 V. @ 10.0 A.	6.0 Lb.	B9906	8.95 ea.
64 or 32 V. @ 8.0 A. ct. & 18 V. @ 8.0 A ct.	10 lb.	B9905	11.95

VOLTAGE REGULATOR BOARDS





B5169 is a board containing 3 15 volt high current regulators with 0.1% regulation. 2 of the regulators are rated @ 3 Amps., and the other @ 6.0 amps. The current in each regulator may be doubled with the regulation going to 0.5%.All 3 regulators are short circuit proof, and 2 have electronic crowbar protection. Brand new, in factory boxes.

STOCK NO.B5169 $11.95 ea. 2/21.00

B9013 is a triple regulator with ± 12 volt regulation @ 200 ma. and the third regulator is a tracking regulator, providing regulation from 0 to 5 volta @ .5 A.

STOCK NO.B9013 $5.95 ea. 2/10.00

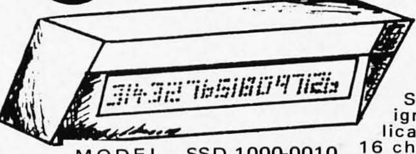
Both regulators above come withcircuit diagrams.

OPERATIONAL AMPLIFIERS
(OP —AMPS)

TYPE	DESCRIPTION	CASE	STOCK	PRICE
709	Hi Performance	TO-5	B4301	.50 5/2.00
4709	Dual 709	DIP	B5301	1.00 6/5.00
741	Hi Performance	DIP	B4316	.65 5/3.00
747	Dual 741	DIP	B4317	1.25 5/5.00
741	Hi Performance	Mini DIP	B4345	.65 5/3.00
747CT	Dual 741	TO-5	B3111	1.25 5/5.00
1458	Dual 741	Mini DIP	B3112	1.25 5/5.00
LM101A	Gen, Purpose	TO-5	B4503	.50 5/2.00

SELF SCAN PANEL DISPLAY

Burroughs



MODEL SSD 1000-0010

BURROUGHS SELF SCAN display, designed for numeric application, requiring up to 16 characters of numeric information. Display is made up of neon dot matrix. Each character is defined by a positive logic 4 bit code. Display operates in a scanning mode, scanning from left to right, one column at a time. Electronics is in interior of bezel, and consists of LSI chip and integrated circuits. Current distributor price is $135.00 . LIMITED QUANTITY

STOCK NO.5180 with data $49.50 2/90.00

Please include sufficient postage. Excess refunded MINIMUM ORDER $5.00



DELTA ELECTRONICS CO.

BOX 1, LYNN, MASSACHUSETTS 01903
Phone (617) 388-4705

Send for the latest edition of our catalog. Loaded with electronic and computer bargains.

BYTE reader service

To get further information on the products advertised in this issue of BYTE merely tear, rip, or snip out this advertiser index, fill out the data at the bottom of the page, mark the appropriate boxes, and send the works to BYTE, Peterborough NH 03458. Readers get extra Brownie Points for sending for information since this encourages advertisers to keep using BYTE – which in turn brings you a bigger BYTE.

ADVERTISER INDEX

Messages for the editor:

Reader's Service
BYTE
Green Publishing Inc.
Peterborough NH 03458

Please print or type.

NOVEMBER 1975

BYTE acquired via
☐ Subscription
☐ Newsstand
☐ Stolen

Name ______________________

Address ______________________

City ______________ State ________ Zip ________

Coupon expires in 60 days . . .

from page 5

at the same time simultaneously so I need to have a serial interface doohinky instead of a parallel doohinky. Luckily MITS has just such a gadget for the Altair, the 88-SIOC, it is called. Once I get this magic board plugged into the Altair I can install a jack to plug in a Teletype. It'll make me feel a lot better.

Not that this is any put down of the Altair for I notice that for some odd reason computers have historically been built without terminals all over the back panel. Let me provide some sage advice to computer designers: Put connectors all over the back panel and make your equipment look like other electronic gear. Stop already with the mystique that computers are something holy and different. They are full of busy electrons like anything else and they should look right.

NEED FOR STANDARDS

The growth of computer systems should be facilitated by some agreement between hobbyists and industry on standards of interface and control. The day may soon arrive when computers are built with standardized connectors in the back for plugging in serial or parallel fed teletype units, television typewriters, cassette recorders, modems, floppy disk drives, etc.

If BYTE can help with this, so much the better. BYTE hereby asks all interested parties to submit papers in support of their proposals for interconnection standards for I/O and memory storage devices. Agreement is important on the standards of interconnection and control, as well as the type of connector to be used. Speak now or hold your peace.

Classified Ads Available for Individuals and Clubs

Readers who have equipment to buy, sell or swap should send in a clearly typed or printed notice to that effect. Insertions should be limited to approximately 100 words or less. Advertisements for this feature can be accepted from individuals or bona-fide clubs of computer users only. Commercial advertisers should contact Bill Edwards, BYTE advertising manager, for the latest rate card and terms. Individual/club classifieds will be printed on a space-available basis in the earliest possible issue of BYTE.



At last! High quality, laboratory-grade test instruments . . . for the professional and hobbyist . . . at prices everyone can afford!

DESIGN MATE 1

CIRCUIT DESIGNER

Now you can build/test electronic circuits WITHOUT SOLDER . . . using solid #22 AWG wire to interconnect discrete components . . . resistors, transistors, linear/digital ICs in TO5 or DIP packages (8-40 pins), and more. Plus, you get 5-15VDC up to 600ma (9 watts) of variable regulated power, with a built-in 0-15V voltmeter to monitor internal power and/or external circuits. Now, that's design flexibility! And look at the low, low price!

49.95
Add $2.50 shipping/handling



SPECIFICATIONS
Power Supply: Output: 5-15V @ 600ma. **Ripple and Noise:** less than 20mv @ full load. Load/Line regulation: <1%. Meter: 0-15V DC. **Connectors:** 1 QT-59S, 2 QT-59B, 2 power supply 5-way binding posts, 2 meter 5-way binding posts. **Wght:** 3 lbs. **Power Needed:** 117V, AC @ 60 Hz 12W.*
Patent #235,554.

64.95



Add $2.50 shipping/handling

DESIGN MATE 2

FUNCTION GENERATOR

Troubleshooting? Design Testing? DM-2 gives you all the signal source capacity you need . . . at a very modest price. This 3-wave form Function Generator has: short-proof output, variable signal amplitude and constant output impedance. Completely wired, tested, calibrated, ready to test audio amplifiers, op-amp and educational lab designs . . . as well as complex industrial lab projects. Complete with easy-to-read instructions/operations manual, application notes, operation theory and more, DM-2 works hand-in-hand with DM-1 for total versatility.

SPECIFICATIONS
Frequency Range: 1Hz-100KHz (5 ranges: 1-10Hz, 10-100Hz, 100-1000Hz, 1-10KHz, 10-100KHz). **Dial Accuracy:** Calibrated @ 10Hz, 100Hz, 1KHz, 10KHz, freq. accurate to 5% of dial setting. **Wave Forms:** Sine <2% THD over freq. range. **Triangle** wave linearity, <1% over range. **Square** wave rise/fall <0.5 microseconds — 600Ω-20pf termination. **Output Amplitude:** (all wave forms) variable-0.1V-10V peak to peak into open circuit. **Output Impedance:** 600Ω-constant over ampl./freq. ranges. **Wght.:** 2 lbs. **Power Needed:** 117V, AC @ 60Hz 5W.*

DESIGN MATE 3

R/C BRIDGE

Have you been bugged by color codes or unreadable component markings? Forget it! DM-3, the low cost R/C Bridge, measures true component values . . . in seconds . . . to better than 5%. And, it's all done with only 2 operating controls and a unique solid-state null detector, to zero-in on exact component selection . . . instantly! Completely wired, calibrated and tested, DM-3 includes an extensive instruction/applications manual, and operational theory too.

SPECIFICATIONS
Resistance Range: 10Ω-100 megΩ. (6 Ranges: 10-100Ω, 100-1000Ω, 1K-10KΩ, 100K-1 megΩ, 1 megΩ-10 megΩ) **Capacitance Range:** 10pFd-1mFd- (5 Ranges: 10-100 pFd, 100-1000pFd, .001-.01 mFd, .01mFd-.1mFd, .1- 1.mFd.) **Null Detector:** 2 hi-intensity LEDs-hi/lo markings. **Accuracy:** <5% of null dial, range switch setting. **Wght.** 2 lbs. **Power Needed:** 117V, AC @ 60Hz 3W.*

54.95
Add $2.50 shipping/handling

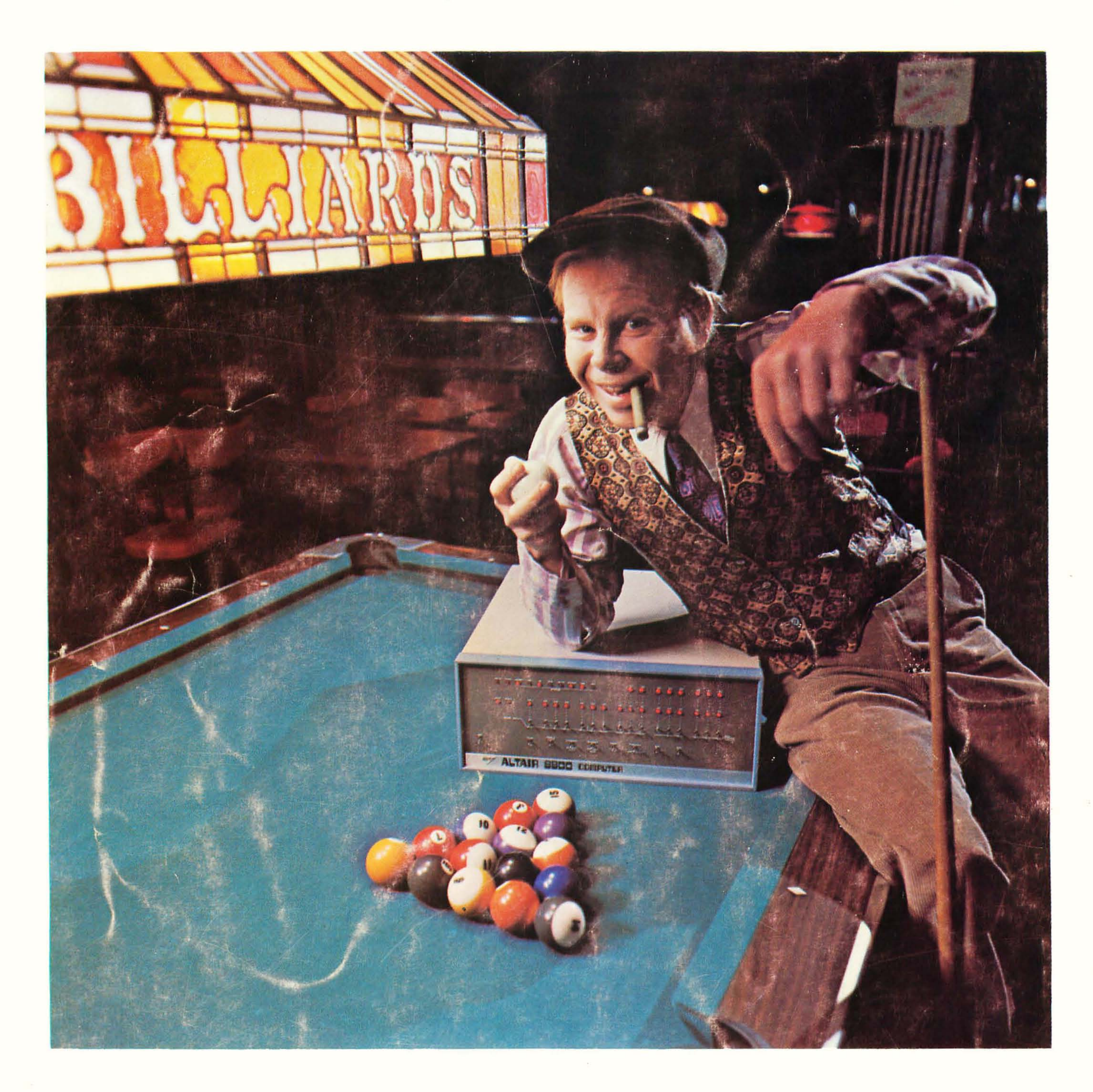


Each measures 6.75''L x 7.5''W x 3.25'' H.; completely assembled, ready to start testing at once. Order your DESIGN MATES today!
**220V @ 50Hz available at slightly higher cost.*

All DESIGN MATES are made in USA; available off-the-shelf from your local distributor. Direct purchases from CSC can be charged on Bank Americard, Master Charge, American Express. Plus, you get a FREE English/Metric Conversion Slide Rule with each order. Foreign orders please add 10% for shipping/handling. Prices are subject to change.

CSC
CONTINENTAL SPECIALTIES CORPORATION
44 Kendall Street, Box 1942, New Haven, CT 06509 • 203/624-3103
West Coast Office: Box 7809, San Francisco, CA 94119 • 415/421-8872
Canada: Len Finkler Ltd., Ontario

The MITS Altair 8800.



(It's showing up in some of the most unusual places.)